TAMPERE UNIVERSITY OF TECHNOLOGY

JASON LLOYD-PRICE

SIMULATING STOCHASTIC CHEMICAL KINETICS WITH DYNAMIC COMPARTMENTALIZATION AT RUNTIME

Master of Science Thesis

Examiners: Olli Yli-Harja
Andre S. Ribeiro
Examiners and subject approved in the Faculty of Computing and Electrical Engineering Council meeting on 9.3.2011

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
JASON LLOYD-PRICE: Simulating stochastic chemical kinetics with dynamic compartmentalization at runtime
Master of Science Thesis, 51 pages
June 2011
Major: Computational Systems Biology
Examiners: Olli Yli-Harja, Andre S. Ribeiro
Keywords: Stochastic Simulation Algorithm, dynamic compartments, λ phage, coupled transcription and translation, logarithmic complexity, prokaryotic gene expression

Stochastic chemical simulations have proven to be a powerful tool to investigate the dynamics of biological systems. Traditional simulations built upon the Stochastic Simulation Algorithm ignore the identity and spatial relationship between the molecules, and instead treat them as being in a homogeneous mixture. However, cells are not homogeneous compartments and their function requires spatial compartmentalization of processes. Space can be introduced into stochastic simulations by statically compartmentalizing the reactants with diffusion reactions between them. Here, we describe a new exact stochastic chemical kinetics simulator which is capable of simultaneously simulating the chemical kinetics within a set of interacting compartments, each of which can be created or destroyed at runtime. The simulator is constructed to be efficient, taking worst-case logarithmic time per reaction and per compartment for each reaction step performed. We then construct and perform a cursory analysis of the dynamics of two models that would not have been possible to simulate at the molecular level without this simulator: a model of a growing/shrinking population of bacteria infected by a bacteriophage, and a single-nucleotide model of coupled transcription and translation in prokaryotes.

PREFACE

This Master's thesis was carried out between 2010-2011 at the Department of Signal Processing, Tampere University of Technology, Tampere, Finland.

I would like to thank Jarno Mäkelä and Abhishekh Gupta for assistance in the modeling aspects of the thesis, and Ilya Potapov for repeatedly finding ways to break the simulator.

I would also like to extend my gratitude to Asst. Prof. Andre S. Ribeiro and Prof. Olli Yli-Harja, for supervising this thesis and without whom this work would not have been possible.

Tampere, May 20th, 2011

Jason Lloyd-Price

CONTENTS

ABBREVIATIONS AND SYMBOLS

CFPE	Chemical Fokker-Panck Equation
CLE	Chemical Langevin Equation
CME	Chemical Master Equation
DM	Direct Method
DNA	Deoxyribonucleic Acid
DSSA	Delayed Stochastic Simulation Algorithm
FFM	First Family Method
FRM	First Reaction Method
LDM	Logarithmic Direct Method
mRNA	Messenger Ribonucleic Acid
NRM	Next Reaction Method
$O(\cdot)$	Asymptotic Upper Bound
ODE	Ordinary Differential Equation
ODM	Optimized Direct Method
RNA	Ribonucleic Acid
RNAp	Ribonucleic Acid Polymerase
RRE	Reaction Rate Equation
SDE	Stochastic Differential Equation
SDM	Sorting Direct Method
SSA	Stochastic Simulation Algorithm
$\Theta(\cdot)$	Asymptotic Tight Bound
$\Omega(\cdot)$	Asymptotic Lower Bound

1. INTRODUCTION

The modeling and simulation of complex biochemical systems has become an important tool to study their behaviors[14, 1, 46, 30]. One biological process that has received a lot of focus in this regard is gene expression, as it is a dynamic process that involves a multitude of components interacting towards a common aim that are not easily separable experimentally. Modeling the process allows the study of the effects of each component independently. Of the modeling techniques for gene expression dynamics, the one that has had the most success in mimicking the dynamics, while remaining computationally tractable, is the Monte Carlo simulation of the Chemical Master Equation (CME).

Stochasticity intrinsic to the process of gene expression has been shown to be non-negligible [5, 41] and an inherent component of the cell's processes. This stochasticity has downstream effects in organisms' phenotypes[1, 3, 21, 22, 37, 40], and is realistically captured by simulations based on the CME. However, the CME makes the assumption that there is no spatial organization between the molecules in the system. Cells are spatially organized compartments and both the time and the place where events and processes take place can affect the phenotype. This constitutes a major problem in stochastic simulations. For example, in prokaryotic gene expression, several RNA polymerases can be transcribing a gene at a given time. Each of the elongating RNA molecules is open to translation. To simulate the movement of ribosomes along the RNA strands, each strand must be distinguishable (that is, the RNA nucleotides cannot be considered to be well-stirred), since otherwise the ribosomes could 'overtake' one another on the template. The simulator must therefore be able to distinguish between a changing number of different strands.

Another common problem when simulating gene expression at a detailed level is the immense number of reactions, as well as reactants, each of which needs to be defined prior to the start of the simulation. Most of these are only needed or present for very short and rare periods of time. For example, each mRNA can consist of thousands of nucleotides. As the RNA polymerase moves along the RNA strand, only a few different reactions can take place at any given time. To simulate these systems, a simulator must either introduce and remove these reactions at runtime[20], or be built to handle large amounts of reactions simultaneously[8].

This thesis proposes a new simulator of stochastic chemical kinetics where the

spaces in which the chemical reactions occur, and the reactants themselves, can be created or removed during the simulation. This allows the simulation of complex biochemical processes (for example the simultaneous translation of an RNA by multiple ribosomes) that would not have been possible with previous simulators. The simulator is constructed with efficiency in mind, so that simulations with thousands of reactions, reactants and compartments can run in a reasonable amount of time.

We begin by describing the relevant aspects of the stochastic framework of chemical kinetics, notably, the Chemical Master Equation. We then present the Stochastic Simulation Algorithm (SSA), a Monte Carlo algorithm to exactly sample the CME. Two recent optimizations and one extension of the SSA are then presented. These augmented algorithms are then used as building blocks to construct the simulator capable of simulating systems of reacting chemicals in compartments that are created and destroyed at runtime. The construction rules and data structures used are presented along with analyses of the runtime complexity. To guarantee that each reaction step is performed in logarithmic time, a new data structure is presented. Finally, the new simulator is applied to biological models which could not have been simulated in a traditional simulation environment; specifically, a model of a population of bacteria infected by a virus, and a detailed model of prokaryotic gene expression.

2. BACKGROUND

2.1 Reaction Rate Equations

The traditional way to describe and simulate the behavior of a homogeneous mixture of chemically reacting molecules is to solve a system of coupled Ordinary Differential Equations (ODEs). The system of ODEs will have one equation for each of the N chemical species that are active in the volume. Each equation describes the 'rate of change' of the concentration X_i of each chemical species S_i, given the concentrations of the other species and the stoichiometry and reaction constants of the R channels through which they interact. This formulation is therefore also referred to as the Reaction Rate Equation[14] (RRE).

In RREs, the vector of concentrations of the molecules $\mathbf{x} = (X_1(t), ..., X_N(t))$ evolves as a continuous-valued function of time of the form[14]:

$$d\mathbf{x} = \sum_{\mu=1}^{R} \nu_\mu r_\mu (\mathbf{x})\, dt \tag{2.1}$$

Here, ν_μ is a vector describing the stoichiometry of reaction $\mathcal{R}_\mu$ and r_μ is the mean 'rate' at which reaction $\mathcal{R}_\mu$ occurs as a function of the current concentration of all chemical species.

This deterministic formulation adequately describes the expected behavior of the chemical system in most cases, but the approximation fails in several situations such as when there are correlations between the concentrations of the molecular species, and when single-molecule events play an important role in the dynamics of the system. In the former case, correlations between the molecular species cause $\langle X_i X_j \rangle$ to not necessarily equal $\langle X_i \rangle \langle X_j \rangle$, leading to an incorrect mean 'rate' in equation (2.1). In the latter case, the effectively random collisions between molecules render the system unpredictable, since the exact timing of the occurrences of a single reaction depends on many factors beyond the concentration of the various molecules in the reaction volume. To include this randomness in RREs, a stochastic term has generally been added to each equation, resulting in a system of Stochastic Differential Equations (SDEs). When the stochastic term is zero-mean Gaussian noise scaled by the reaction rates, the system of equations becomes what is known as the Chemical

Langevin Equation[13] (CLEs), which takes the form:

$$d\mathbf{x} = \sum_{\mu=1}^{R} \nu_\mu r_\mu(\mathbf{x})\, dt + \sum_{\mu=1}^{R} \nu_\mu \sqrt{r_\mu(\mathbf{x})} \mathcal{N}_\mu(0,1)\sqrt{dt} \tag{2.2}$$

where each $\mathcal{N}_\mu(0,1)$ is a statistically independent normally distributed random variable.

The state $\mathbf{x}$ is no longer uniquely determined by the initial state of the system $\mathbf{x}_0$ at $t = t_0$, and instead follows the probability density function $P(\mathbf{x}, t|\mathbf{x}_0, t_0)$. This N-dimensional probability distribution obeys a well-defined partial differential equation called the Chemical Fokker-Planck Equation[12] (CFPE).

Nevertheless, due to the use of continuous variables for $\mathbf{x}$, RREs and CLEs (and by extension CFPEs) have difficulty in correctly capturing the fluctuations that are intrinsic to the discrete nature of chemical reactions. In systems where the appearance or disappearance of a single molecule is significant to the behavior of the system, these approximations do not reproduce the correct dynamics. Unfortunately, such conditions are ubiquitous in biological models, where many pathways include steps which involve low numbers of molecules[1, 5, 41].

2.2 The Chemical Master Equation

Another approach to the kinetics of a spatially homogeneous system of reacting chemicals which captures these low copy number effects is the stochastic formulation[26]. Here, the state vector $\mathbf{x}$ represents the absolute number of molecules in the reaction volume at time t, and ν_i represents the absolute number of each reactant that change when reaction $\mathcal{R}_i$ occurs. The reaction constants are no longer viewed as rates of occurrence, but rather as probabilities of occurrence per unit time. The time-evolution of $\mathbf{x}$ then takes the form of a random walk through the N-dimensional space of the populations of the reacting species.

At heart, the stochastic formulation of chemical kinetics is based on the propensity function a_μ, which is defined as:

$$a_\mu(\mathbf{x})\, dt \equiv \text{the probability that a particular combination of the reactants that are presently in the system will react via reaction } \mathcal{R}_\mu \text{ in the next infinitesimal time interval } [t, t+dt). \tag{2.3}$$

From this definition alone, the master equation for a chemical system can be derived. It can thus be considered to be the fundamental premise of the stochastic formulation of chemical kinetics. The form that the function a_μ takes depends on the type of the reaction it represents.

Unimolecular reactions represent processes that are internal to each molecule of a given molecular species. These processes are usually quantum mechanical in nature, and can only be described by their probability of occurrence. For example, let reaction $\mathcal{R}_\mu$ be a possible change in the properties of a molecule of species S_i, common to all molecules of this species. This implies that for any given molecule of S_i, there is some constant c_μ such that $c_\mu dt$ is the probability that that molecule will spontaneously react via $\mathcal{R}_\mu$ in the next infinitesimal time dt. It follows that if there are currently X_i molecules of S_i currently in the system, the probability that one of them will react via $\mathcal{R}_\mu$ in the next infinitesimal time interval is $X_i c_\mu dt$. The propensity function for unimolecular reactions is therefore $a_\mu(\mathbf{x}) = X_i c_\mu$.

Bimolecular reactions involving two different molecular species S_i and S_j occur when two such molecules meet and react. Using the homogeneous assumption, it has been shown that there exists a constant c_μ such that $c_\mu dt$ gives the probability that a given *pair* of molecules will meet and react via reaction $\mathcal{R}_\mu$[10]. This constant can be derived from microphysical properties[11], giving the bimolecular propensity function an *a priori* claim to validity that the reaction rate constants do not have. If there are X_i molecules of S_i and X_j molecules of S_j currently in the system, then there are $X_i X_j$ pairs of these molecules. The probability that one of these pairs will meet and react via $\mathcal{R}_\mu$ in the next infinitesimal time dt is therefore $X_i X_j c_\mu dt$. The propensity function for bimolecular reactions with different chemical species is then $a_\mu(\mathbf{x}) = X_i X_j c_\mu$.

For bimolecular reactions between two molecules of the same species S_i, the number of pairs does not grow as X_i^2, since a molecule cannot react with itself. Instead, the number of pairs grows as $X_i(X_i - 1)/2$, making the propensity function for such a reaction $a_\mu(\mathbf{x}) = X_i(X_i - 1)c_\mu/2$.

Trimolecular reactions and above are not realistically possible, but can be used as simplified approximations of a series of bimolecular reactions. The number of combinations of the reactants in such reactions is calculated from the total number of combinations of the reactants without replacement. Their propensity functions are given here for completeness, and for some specific model constructions, but should not be considered to be grounded directly in molecular kinetics[9].

To summarize the above, the propensity function of reaction $\mathcal{R}_\mu$ is $a_\mu(\mathbf{x}) = h_\mu(\mathbf{x})c_\mu$, where $h_\mu(\mathbf{x})$ gives the number of possible reactant combinations in the reaction volume. $h_\mu(\mathbf{x})$ for different reaction templates is given in Table 2.1. Here, $\mathcal{S}_\mu$ refers to the set of molecular species which are reactants in reaction μ, and $N_{(i,\mu)}$ represents the number of molecules of species S_i that are consumed by reaction $\mathcal{R}_\mu$.

In the stochastic formulation, we would like to describe the probability $P(\mathbf{x}, t|\mathbf{x}_0, t_0)$ of having a given number of each reactant in the reaction volume at time t after the initial conditions $\mathbf{x} = \mathbf{x}_0$ at $t = t_0$. Using equation (2.3), the rate of change

Reaction Type	$h_\mu(\mathbf{x})$
$\longrightarrow$ products	1
$S_i \longrightarrow$ products	X_i
$S_i + S_j \longrightarrow$ products, $i \neq j$	$X_i X_j$
$2S_i \longrightarrow$ products	$\frac{X_i(X_i-1)}{2}$
$\sum_{i\in\mathcal{S}_\mu} N_{(i,\mu)} S_i \longrightarrow$ products	$\prod_{i\in\mathcal{S}_\mu}\prod_{q=1}^{N_{(i,\mu)}} \frac{X_i - q + 1}{q}$

Table 2.1: The number of possible combinations $h_\mu(\mathbf{x})$ in which a given reaction $\mathcal{R}_\mu$ can occur, given the type of reaction that $\mathcal{R}_\mu$ is.

of the probability of being in a given state $\mathbf{x}$ can be expressed as the sum of the probabilities of all reactions that can change the system's state into $\mathbf{x}$ in the next infinitesimal time interval, subtracting the sum of the probabilities of all reactions that can cause the system to *leave* that state. The result is a partial differential equation for P, the Chemical Master Equation[26] (CME):

$$\frac{\partial P(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t} = \sum_{\mu=1}^{R} \left[a_\mu(\mathbf{x} - \nu_\mu) P(\mathbf{x} - \nu_\mu, t|\mathbf{x}_0, t_0) - a_\mu(\mathbf{x}) P(\mathbf{x}, t|\mathbf{x}_0, t_0)\right] \quad (2.4)$$

The function that satisfies the CME simultaneously describes the probability of all possible trajectories through the N-dimensional state space of reactant populations. This function can be seen as the discrete analogue to the CFPE. Because of the explicit handling of every possible state that the system can be in, the CME can take an accurate account of the effects of both fluctuations and correlations. This has been a major justification for using the stochastic approach over the mathematically simpler deterministic approach.

2.3 The Stochastic Simulation Algorithm

Analytically solving RREs or the CME is an intractable problem for systems of more than a few reactants and reactions[9]. However, considerable success has been realized in numerical simulations of these systems. For RREs and CLEs, this implies the use of numerical integration techniques. For the CME, this requires the use of a Monte Carlo method to numerically simulate the Markov process that the CME describes.

The Stochastic Simulation Algorithm (SSA) is a method to exactly sample trajectories of the CME. That is, it provides a new time series of $\mathbf{x}$ every time it is run, with each possible trajectory appearing with exactly the frequency that the CME prescribes. Instead of solving the partial differential equations in (2.4), the SSA is

based on the probability distribution defined as follows:

$$\mathcal{P}(\tau, \mu|\mathbf{x})d\tau \equiv \text{the probability that the next reaction in the system will be } \mathcal{R}_\mu, \text{ and it will occur in the infinitesimal time interval } [t+\tau, t+\tau+d\tau). \tag{2.5}$$

where $0 \leq \tau < \infty$ and $\mu \in 1, ..., R$.

All exact formulations of the SSA follow the following general procedure[9]:

0. **Initialization**: Set time $t = 0$. Set up the initial state vector $\mathbf{x} = \mathbf{x}_0$. Calculate and store the values of the propensities of each reaction, $a_i, i \in \{1..R\}$.
1. **Selection**: Using a suitable sampling procedure, generate a random pair (τ, μ) according to the joint probability distribution in equation (2.5).
2. **Sampling**: Output the system state for each sampling point in the time interval $[t, t+\tau)$.
3. If $t + \tau \geq t_{stop}$, terminate.
4. **Execution**: Set $t = t + \tau$, and $\mathbf{x} = \mathbf{x} + \nu_\mu$.
5. **Update**: Recalculate a_i for all i such that any X_j that was changed in step 4 appears as a reactant in $\mathcal{R}_i$.
6. Go to step 1.

One iteration of the loop, from step 1 to step 6, is here called a "reaction step". Runtimes of all algorithms are evaluated in terms of their runtime per reaction step, since this loop is unavoidable in any implementation of the exact SSA.

The procedure in step 1 can be accomplished with two statistically equivalent sampling procedures, which give rise to the two original formulations of the SSA[9]: the First Reaction Method (FRM) and the Direct Method (DM).

2.3.1 First Reaction Method

The FRM is the mathematically simplest, but most computationally intensive method to sample the joint distribution in equation (2.5). The idea is to generate a set of tentative firing times τ_v for each reaction channel, and pick μ based on those. For this, we need the probability density function $P_v(\tau|\mathbf{x})$ such that $P_v(\tau|\mathbf{x})d\tau$ gives the probability that reaction $\mathcal{R}_v$ occurs in the infinitesimal time interval $(t+\tau, t+\tau+d\tau)$, assuming that no other reaction occurs in the interval $(t, t+\tau)$.

To calculate $P_v(\tau|\mathbf{x})d\tau$, we first write it as the product of the probability P_0^v that $\mathcal{R}_v$ does *not* occur in the interval $(t, t+\tau)$, and the subsequent probability that it does occur in the interval $(t+\tau, t+\tau+d\tau)$:

$$P_v(\tau|\mathbf{x})d\tau = P_0^v(\tau) \cdot a_v(\mathbf{x})d\tau \tag{2.6}$$

To calculate P_0^v, let us divide the interval $(t, t+\tau)$ into K subintervals of equal duration. From equation (2.3), the probability that $\mathcal{R}_v$ does *not* fire in the first subinterval $(t, t+\tau/K)$ is $1 - a_v(\mathbf{x})\tau/K$. This is also the probability that it will not fire in the second time interval $(t+\tau/K, t+2\tau/K)$, given that it did not fire in the first, and so on. P_v can then be written as:

$$P_0^v(\tau|\mathbf{x}) = (1 - a_v(\mathbf{x})\tau/K)^K \tag{2.7}$$

Taking the limit as K goes to infinity and applying the limit formula for the exponential function, we get:

$$P_0^v(\tau|\mathbf{x}) = e^{-a_v(\mathbf{x})\tau} \tag{2.8}$$

Substituting (2.8) into (2.6):

$$P_v(\tau|\mathbf{x})d\tau = a_v(\mathbf{x})e^{-a_v(\mathbf{x})\tau}d\tau \tag{2.9}$$

P_v is the well-known exponential distribution with rate parameter $\lambda = a_v(\mathbf{x})$, which can be sampled from a uniform random number U in the range $(0, 1)$ with the formula[9]:

$$\tau_v = \frac{-\ln U}{a_v(\mathbf{x})} \tag{2.10}$$

The set of tentative firing times can then be obtained by applying (2.10) for each possible reaction. We then select the actual reaction to perform and the actual firing time (τ, μ) as:

$$\mu = v \mid \tau_v < \tau_i \quad \forall i : i \neq v \tag{2.11}$$

$$\tau = \tau_\mu \tag{2.12}$$

2.3.2 Direct Method

The Direct Method generates the pair (τ, μ) directly from the joint probability density function in equation (2.5). Let $a_0(\mathbf{x})dt$ be the probability that any of the possible reactions occurs in the system in the next infinitesimal time interval $(t, t+dt)$. Since at most one reaction can occur in an infinitesimal time interval, a_0 is the sum of the

individual probabilities that each reaction will occur:

$$a_0(\mathbf{x})dt = \sum_{v=1}^{R} a_v(\mathbf{x})dt \tag{2.13}$$

Similar to (2.6), we can now write $\mathcal{P}(\tau, \mu|\mathbf{x})$ explicitly as the probability $\mathcal{P}_0(\tau|\mathbf{x})$ that *no* reaction occurs in the interval $(t, t+\tau)$, multiplied with the probability that reaction $\mathcal{R}_\mu$ occurs in the following infinitesimal time interval $d\tau$:

$$\begin{aligned} \mathcal{P}(\tau, \mu|\mathbf{x})d\tau &= \mathcal{P}_0(\tau|\mathbf{x}) \cdot a_\mu(\mathbf{x})d\tau \\ &= e^{-a_0(\mathbf{x})\tau} \cdot a_\mu(\mathbf{x})d\tau \end{aligned} \tag{2.14}$$

To turn this distribution into one for which we can generate random numbers, we condition $\mathcal{P}(\tau, \mu|\mathbf{x})$ in the form[9]:

$$\mathcal{P}(\tau, \mu|\mathbf{x}) = \mathcal{P}_1(\tau|\mathbf{x}) \cdot \mathcal{P}_2(\mu|\tau, \mathbf{x}) \tag{2.15}$$

where $\mathcal{P}_1(\tau)d\tau$ is the probability that the next reaction, whichever it might be, will occur in the interval $(t+\tau, t+\tau+d\tau)$, while $\mathcal{P}_2(\mu|\tau, \mathbf{x})$ is the probability that this reaction is $\mathcal{R}_\mu$, given that it occurs in that interval.

$\mathcal{P}_1(\tau|\mathbf{x})d\tau$ is the marginal distribution of $\mathcal{P}(\tau, \mu|\mathbf{x})d\tau$, which can be obtained by summing over all μ:

$$\mathcal{P}_1(\tau|\mathbf{x}) = \sum_{v=1}^{R} \mathcal{P}(\tau, v|\mathbf{x}) \tag{2.16}$$

Substituting (2.16) into (2.15) and solving for $\mathcal{P}(\mu|\tau, \mathbf{x})$ yields:

$$\mathcal{P}_2(\mu|\tau, \mathbf{x}) = \frac{\mathcal{P}(\tau, \mu|\mathbf{x})}{\sum_{v=1}^{R} \mathcal{P}(\tau, v|\mathbf{x})} \tag{2.17}$$

Together, equations (2.16) and (2.17) express the joint probability distribution (2.14) as two univariate distributions. Substituting (2.14) into both of them gives:

$$\mathcal{P}_1(\tau|\mathbf{x}) = a_0(\mathbf{x})e^{-a_0(\mathbf{x})\tau} \tag{2.18}$$

$$\mathcal{P}_2(\mu|\tau, \mathbf{x}) = \frac{a_\mu(\mathbf{x})}{a_0(\mathbf{x})} \tag{2.19}$$

Equation (2.18) follows an exponential distribution with rate parameter $\lambda = a_0(\mathbf{x})$. The time until the *next* reaction, whatever it may be, can now be randomized similar to (2.10). The distribution in equation (2.19) is a multinomial distribution with each μ having probability a_μ/a_0. Note that it is in fact independent from τ. A correctly distributed random μ can then be generated from a uniform random

number U in the range $[0, 1)$ by finding v such that:

$$\begin{aligned}
\sum_{i=1}^{v-1} \mathcal{P}_2(v|\tau, \mathbf{x}) \leq \quad & U \quad \leq \sum_{i=1}^{v} \mathcal{P}_2(v|\tau, \mathbf{x}) \\
\sum_{i=1}^{v-1} \frac{a_\mu(\mathbf{x})}{a_0(\mathbf{x})} \leq \quad & U \quad \leq \sum_{i=1}^{v} \frac{a_\mu(\mathbf{x})}{a_0(\mathbf{x})} \\
\sum_{i=1}^{v-1} a_\mu(\mathbf{x}) \leq U a_0(\mathbf{x}) & \leq \sum_{i=1}^{v} a_\mu(\mathbf{x})
\end{aligned} \tag{2.20}$$

The final pair (τ, μ) is then randomized based on (2.20) and (2.16) from two independent uniform random numbers U_1 and U_2 with respective ranges $[0, 1)$ and $(0, 1)$ as follows:

$$\begin{aligned}
\mu &= v \mid \sum_{i=1}^{v-1} a_i(\mathbf{x}) \leq U_1 a_0(\mathbf{x}) < \sum_{i=1}^{v} a_i(\mathbf{x}) \\
\tau &= \frac{-\ln U_2}{a_0(\mathbf{x})}
\end{aligned} \tag{2.21}$$

2.4 Improvements to the SSA

While the SSA can be used to sample the CME of a chemical system considerably larger than what is accessible by directly integrating the CME, it too becomes computationally prohibitive for even moderate system sizes. This problem is compounded by the fact that a single trajectory generated by the SSA is rarely used on its own - the kinetics of the stochastic chemical systems are often described by distributions gained by repeatedly running the SSA. Hence, some improvements have been made to the algorithm to improve its computational efficiency without affecting its exactness.

2.4.1 Next Reaction Method

The Next Reaction Method[8] (NRM) attempts to reduce the computational cost of the FRM by reusing the tentative times generated in previous iterations and storing them in a special data structure, an Indexed Priority Queue. The FRM as described in section 2.3.1 contains two steps which run in linear time on the number of reactions: generating the R tentative reaction times, and selecting the smallest tentative reaction time. These will both be addressed by the NRM. To avoid confusion of notation, the current simulation time t is written as t_1 in this section.

Let us consider the execution of the first iteration of the FRM. Prior to the

Execution step, the distribution of $t_v = t_1 + \tau_v$ follows the distribution of $P_v(t|t_1, \mathbf{x})$ obtainable from equation (2.10):

$$P_v(t|t_1, \mathbf{x}) = H(t - t_1)a_v(\mathbf{x})e^{-a_v(\mathbf{x})(t-t_1)} \tag{2.22}$$

where $H(x)$ is the Heaviside function which is 0 for $x < 0$ and 1 for $x \geq 0$.

During the Selection step, the FRM selects the reaction with the minimum τ_μ. We therefore know that $t_v > t_\mu$ for all $v \neq \mu$. It follows that t_v is distributed according to the conditional distribution:

$$\begin{aligned} P_v(t|t_1, \mathbf{x}, t > t_\mu) &= \frac{H(t - t_\mu)P_v(t|t_1, \mathbf{x})}{P_v(t > t_\mu|t_1, \mathbf{x})} \\ &= \frac{H(t - t_\mu)H(t - t_1)a_v(\mathbf{x})e^{-a_v(\mathbf{x})(t-t_1)}}{e^{-a_v(\mathbf{x})(t_\mu - t_1)}} \\ &= H(t - t_\mu)a_v(\mathbf{x})e^{-a_v(\mathbf{x})(t-t_\mu)} \\ &= P_v(t|t_\mu, \mathbf{x}) \end{aligned} \tag{2.23}$$

During the Execution step, t_1 is advanced to t_μ, implying that the t_v values are *already* correctly distributed for the next reaction step if no propensities change. If propensities do change, the times must be adjusted somehow. During the Execution step, the state will be updated to $\mathbf{x} + \nu_\mu$. The distribution of t_v^{new}, the desired distribution of t_v *after* the Execution step is then:

$$P_v^{new}(t|t_\mu, \mathbf{x}) = H(t - t_\mu)a_v(\mathbf{x} + \nu_\mu)e^{-a_v(\mathbf{x}+\nu_\mu)(t-t_\mu)} \tag{2.24}$$

The transformation from the previous distribution (equation (2.23)) to the distribution that we want (equation (2.24)) can be obtained by setting their cumulative distribution functions equal to each other:

$$\begin{aligned} \int_{-\infty}^{t_v^{new}} P_v^{new}(t|t_\mu, \mathbf{x})dt &= \int_{-\infty}^{t_v} P_v(t|t_\mu, \mathbf{x})dt \\ 1 - e^{a_v(\mathbf{x}+\nu_\mu)(t_v^{new} - t_\mu)} &= 1 - e^{a_v(\mathbf{x})(t_v - t_\mu)} \\ t_v^{new} &= \frac{a_v(\mathbf{x})}{a_v(\mathbf{x} + \nu_\mu)}(t_v - t_\mu) + t_\mu \end{aligned} \tag{2.25}$$

We can now transform the t_v during the Update step of the SSA so that they follow the new distribution of firing times. Since the t_v values were generated independently and the transformation does not introduce any dependence (it is not based on any other t_v values), the transformed times are also statistically independent. This allows us to reuse almost all of the t_v values from previous iterations, most of which won't even need to be transformed if the propensity of the reaction does not change. There

are only two conditions when new t_v values need to be generated: t_μ itself cannot be reused, and any reaction that had zero propensity in the previous iteration (its t_v will be ∞).

With the cost of generating new t_v values minimized by the reuse of previously generated values, the NRM now only contains one step which runs in linear time: selecting the minimum t_v. The authors introduce a heap-like data structure which they name the Indexed Priority Queue in order to minimize the cost of this operation[8]. This data structure is a binary tree which contains one node for each reaction channel in the system. The nodes are arranged such that each node represents a reaction whose tentative firing time is earlier than both of its daughter nodes. The reaction with the earliest tentative firing time can then be found by examining only the root node, making the Selection step run in constant time.

In order to maintain the ordering of the nodes during the Update step, whenever a modification is made to a t_v (either by rerandomization or by transformation by (2.25)), the node representing that reaction is moved up or down in the tree until the ordering is restored. This operation requires that a two-way mapping exists between reactions and their placement in the binary tree so that when a particular t_v is modified, the appropriate node in the tree can be looked up in constant time.

An example of the indexed priority queue partway through a simulation is shown in Figure 2.1. Ordering of the nodes such that the earliest reaction is at the top of the tree is visible. Of note is the fact that the tree must support infinite firing times for reactions with zero propensity.

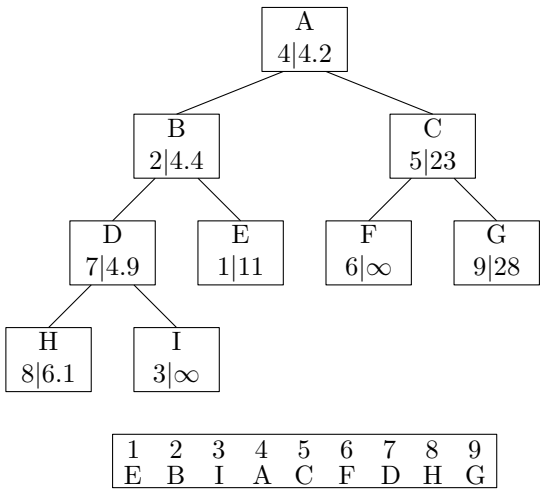

Figure 2.1: Example of the NRM's indexed priority queue[8] partway through a simulation with 9 possible reactions. Reactions are indexed with numbers and nodes are indexed with letters. The nodes of the tree are shown with $v|t_v$. The firing times of each reaction are shown in the node in the tree. The mapping from reactions to nodes in the tree is also shown.

The operations that move nodes in the tree perform at most one operation per

level of the binary tree, making every operation $O(\log R)$. Therefore, if there are M propensities updated in a given reaction step, it follows that the worst-case runtime is $O(M \log R)$. So long as there is no reactant that is common to $\Theta(R)$ reactions, M will vanish with R, and the Update step will run in $O(\log R)$ time.

2.4.2 Logarithmic Direct Method

The NRM is considerably more complex to implement than the DM, and in many small reaction systems, the effort to implement it is not worth the gains. Further, it has been reported that the extra overhead from maintaining the indexed priority queue is a considerably limiting factor in small simulations[2]. Thus, efforts have focused on improving the DM.

Cao and colleagues[2] proposed a modification to the DM, the Optimized Direct Method (ODM), whereby a_0 is stored along with the rest of the propensities. When propensity a_i is recalculated in the Update step, the old value is first subtracted from a_0 and the new value is added, keeping it up to date. a_0 then does not need to be recalculated every reaction step, eliminating an $O(R)$ operation from the Selection step. However, the selection of μ is still an $O(R)$ operation. To minimize the impact of this linear search, the authors proposed to sort the reactions in order of decreasing a_i, allowing the linear search to terminate at an earlier i. Since the propensities can change frequently during the simulation, the authors proposed to set the reaction order by running the simulation for a short period beforehand, and sorting the reactions in descending order of the number of times each reaction fired.

McCollum and colleagues[25] have improved the ODM by removing the pre-run step and instead reordering the reactions at runtime. Every time a reaction occurs, it is swapped with the one below it in the list. The set of high-frequency reactions will then 'bubble up' to the front of the reaction list. This method, dubbed the Sorting Direct Method (SDM), therefore also allows the simulator to adapt to changes in the frequencies of the reactions.

The ODM and the SDM both suffer from two problems. First, despite the reshuffling of reactions, the linear search remains, imposing a worst-case runtime of $O(R)$ on the Selection step. Second, both methods suffer from a precision problem. In a digital computer, there are a limited number of bits that can be carried in the mantissa of a_0. If one reaction has a significantly higher propensity than another reaction, then both the ODM and SDM will place it near the front of the list. The number of significant bits available to store the difference between the sums in equation (2.21) may not be enough to account for the difference between them. Additionally, in simulations where the value of a_0 covers a wide dynamic range over time due to rare reactions with high propensity, a large amount of the bits in the mantissa of a_0 will be lost after the reaction with high propensity has been performed

since the initial addition of its propensity will drop these bits. Another approach has been taken to optimizing the DM which avoids these issues.

The Logarithmic Direct Method (LDM) was proposed[19] (also called the Optimized Direct Method in [8]). In the LDM, the values of the reaction propensities are stored in the leaves of a binary tree. Non-leaf nodes contain the sum of the propensities of their two children. Each non-leaf node then contains a portion of the sum in equation (2.13), and are therefore referred to as partial sums. An example tree of partial sums with eight possible reactions is depicted in Figure 2.2A. For simplicity, this could be implemented as a single array, rather than a linked tree with special treatment for the leaves. In this case, indices below R correspond to partial sums, and indices above or equal to R correspond to propensities. Since this system stores R reaction propensities and $R-1$ partial sums, it still uses $\Theta(R)$ storage space.

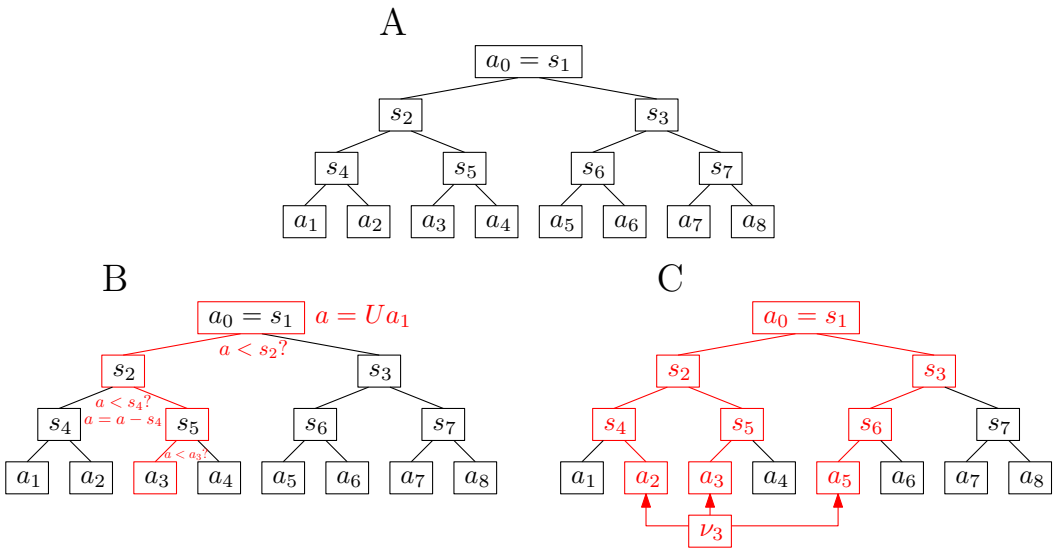

Figure 2.2: (A) Example of a tree of partial sums in the LDM. This system has eight possible reactions with propensities a_1 to a_8. Each partial sum is the sum of all propensities below it in the tree. s_1 is therefore the sum of all propensities, which is a_0. (B) Example of a binary search through the tree of partial sums for the μ that satisfies equation (2.19). Operations at each node are shown in red below it. In this case, $\mu = 3$. (C) Example of the Update step for the LDM. Reaction $\mathcal{R}_3$ changed the state $\mathbf{x}$ to $\mathbf{x} + \nu_3$, which changed the populations of the reactants of reactions $\mathcal{R}_2$, $\mathcal{R}_3$ and $\mathcal{R}_5$. The changed propensities were propagated up the tree of partial sums by recalculating each sum until the root was reached.

During the Selection step, τ can be calculated from a_0, which is readily available at the root of the tree of partial sums. μ is selected by randomizing $a = U_1 a_0$, and then performing a binary search through the tree of partial sums as shown in Figure 2.2B. It follows that the runtime of the Selection step is therefore proportional to the height of the tree, $\Theta(\log R)$.

During the Update step, when a propensity a_i is updated, then all $\log_2 R$ partial sums above it in the tree are also updated. This implicitly updates a_0, which can

then immediately be used in the next reaction step. Figure 2.2C shows this process. If M propensities are updated, then it follows that the runtime of the Update step is $\Theta(M \log R)$. As with the NRM, M will vanish with R, and the runtime of the Update step will be $\Theta(\log R)$.

2.4.3 First Family Method

The First Family Method[14] (FFM) can be seen as a generalization of the above methods. In the FFM, reactions are grouped into 'families'. In the Selection step, instead of choosing a reaction based on the reaction's propensity, the family containing the next firing reaction is selected either by the FRM or the DM, based on the total propensities of the families. μ is then selected from the within the selected family by either the FRM or the DM according to the actual propensities of the reactions in the family.

One advantage of this method is that by grouping many reactions with low propensity together into one family and fewer reactions with high propensity into other families, the FFM can reduce the dynamic range of propensities with which it must deal with at any given moment in the simulation. This reduction reduces the number of bits that are needed in the mantissa to correctly select reactions with the correct distribution if the dynamic range of propensities is very large. If this method is generalized further to allow families within families, the LDM can be seen as a special case of the FFM, where each family contains two reactions or two families.

2.4.4 Delayed SSA

A different type of extension to the SSA was proposed in [8] to simulate complex processes that may take a non-negligible amount of time to complete once initiated. These processes may range from multi stepped processes of simple reactions, such as stepwise elongation, to conformational changes in large structures, such as the unwinding of the DNA, among others. The duration of these processes can be comparable to a cell's lifetime. For example, the elongation of an RNA molecule by an RNA polymerase can take as long as several minutes. The authors showed how these processes could be represented as single-step reactions where its products are released some time later, and not necessarily at the same time. The disadvantage of this method is that the system's evolution in time is no longer a Markov process, and therefore cannot be simulated with any of the above methods as described.

Nevertheless, these lengthy events take place in cells and can cause non-negligible effects in the dynamics of the system, namely, the gene regulatory network[31], and can therefore not be ignored when constructing a model. The exact nature of the process may not be known, so constructing the explicit model may not be

feasible or possible. Allowing reactions that produce their products an arbitrary time later in the simulation provides two advantages. First and foremost, it allows the modeler to insert delays of arbitrary distributions even if the underlying process is not known. Secondly, it removes potentially many reactions from the system, speeding up simulations considerably[8].

To implement reactions that can have delayed products with potentially different delays, these products are placed on a "wait list" as a tuple (t_r, i, n), where t_r is the time at which the product, n molecules of S_i, should be released. The wait list itself can be implemented directly in the NRM with logarithmic time per addition or removal by adding products on the wait list as nodes of the indexed priority queue. Alternatively, it can be implemented to run alongside a DM implementation as a heap-based priority queue with the same runtime bounds using the following variant of the Execution step (paraphrased from [35]):

if $\tau < t_{min}$, where t_{min} in the earliest entry in the wait list **then**
  Perform the normal SSA Execution step
else
  Set $t = t_{min}$, and $X_i = X_i + n$.
  Remove the earliest entry from the wait list.
end if

Reactions with delayed products are represented as:

$$\text{reactants} \longrightarrow a(\tau) + b \tag{2.26}$$

Here, the molecule of species a will be released back into the system τ time after the occurrence of reaction (2.26), whereas the molecule of b is released at exactly the same time as the reaction. τ can follow an arbitrary distribution, but will most often be either Delta (as in reaction (2.26)) or Gamma distributed.

2.5 Compartmentalized Systems

The previous sections have dealt with construction of simulation algorithms to simulate the time-evolution of a spatially homogeneous mixture of a set of chemicals in a single reaction volume. However, many systems are not well-described by a homogeneous mixture of the molecules in the reaction volume. Spatial models in which space is divided into discrete regions called compartments have thus been proposed as a means to correctly capture the dynamics of spatially inhomogeneous distributions of molecules.

In this thesis, we build a simulator to efficiently simulate the chemical kinetics of a set of molecules within a growing or shrinking set of compartments. To illustrate the use of this simulation, and to test its efficiency, we present two models that would

be not possible to implement at this level of detail without dynamic compartments. These models are based on the λ bacteriophage and transcription and translation in prokaryotes.

2.5.1 The λ bacteriophage

The λ bacteriophage is a virus that infects *Escherichia coli* bacteria. Upon infection of a bacterium, the phage enters a pathway termed the "lytic cycle", where it begins replicating itself by hijacking the cell's replication machinery, finally ending in the destruction of the cell and the release of many new virus particles into the environment. An alternate pathway that can be taken is the "lysogenic cycle", in which the phage integrates itself into the bacterial chromosome and lies dormant for some time before eventually reawakening into the lytic pathway[45]. The process of deciding between lysis or lysogeny is a classic example of a stochastic biochemical switch[1].

Arkin and colleagues developed a stochastic kinetic model of this genetic circuit at the molecular level, whose dynamics was simulated by the SSA[1]. They validated the model by comparing it with measurements in variable conditions and then showed that when two independently produced regulatory proteins, acting at low cellular concentrations, competitively control a switch in a pathway, stochastic variations in their concentrations produce probabilistic pathway selection. The result is that an initially homogeneous cell population partitions into distinct subpopulations with different phenotypes[1]. They also showed how the λ-phage uses this lysis-lysogeny decision circuit to randomly switch surface features to evade host responses. Finally, they showed that deterministic kinetics could not predict the statistics of the system due to its probabilistic outcomes.

2.5.2 Transcription and Translation in Prokaryotes

Gene expression is the process by which the DNA in a cell is used as a template to produce proteins, the functional units in the cell. It is composed of two processes: transcription and translation, which respectively use the DNA as a template to produce messenger RNA (mRNA), and use the mRNA to produce proteins. Given how fundamental these two processes are to the function of the cell, it is important to understand their dynamical properties and the impact these properties have on the rest of the cellular processes. Without this understanding, no dynamical description of the cell is possible, and thus no predictive model can be created.

Transcription and translation occur differently in eukaryotes and prokaryotes. Eukaryotic DNA is contained in a delineated nucleus where transcription occurs. The mRNA must be transported into the cell's cytoplasm before it can be translated.

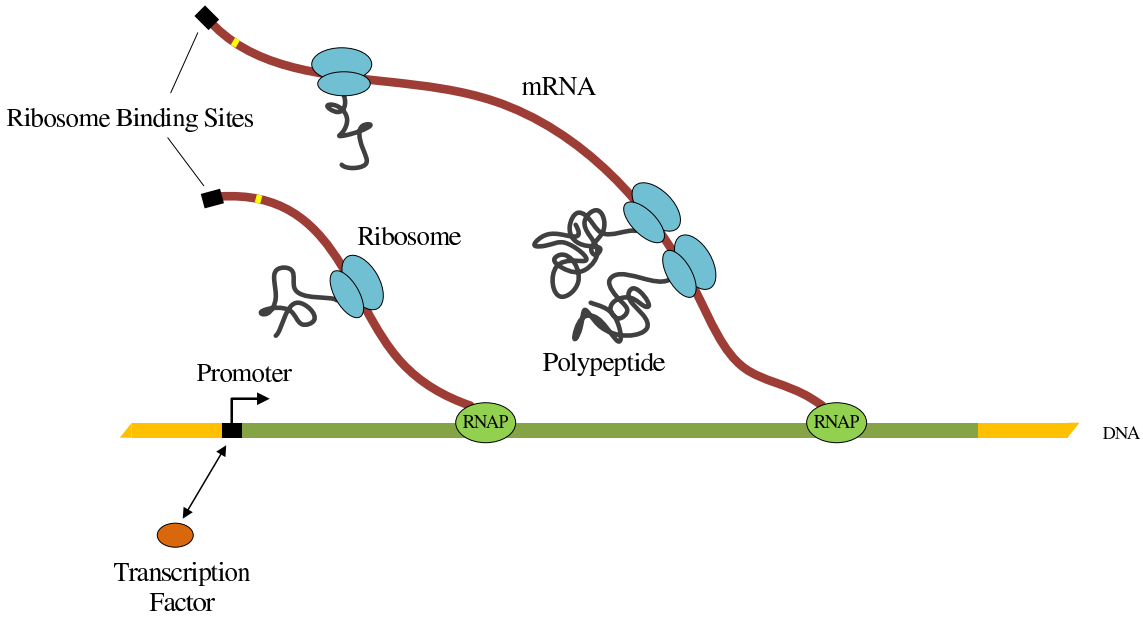

Figure 2.3: Schematic of coupled transcription and translation in prokaryotes

There is no nucleus in a prokaryotic cell, allowing translation to begin immediately after enough of the RNA has been elongated to allow the Ribosome to bind (shown in Figure 2.3). Transcription and translation are therefore said to be *coupled* in prokaryotes.

This coupling causes the propagation of fluctuations from RNA to proteins to differ significantly in prokaryotes and eukaryotes, and likely led to the evolution of different means to regulate them[23]. Several events in prokaryotic transcription and in translation are probabilistic in nature, and their kinetics are sequence dependent. One example is sequence-dependent transcriptional pausing. When they occur, these events can affect the degree of fluctuations of RNA and protein levels [30]. The degree to which protein levels are affected varies from gene to gene as it is largely dependent on how tightly transcription and translation are coupled.

To study the process of gene expression and propagation of fluctuations between RNA and proteins, one needs to be able to stochastically simulate coupled transcription and translation. From the point of view of the actual simulation, the model must allow the different RNA strands in the system to be differentiated, since ribosomal traffic depends on which RNA each ribosome is currently translating. That is, the RNA nucleotides cannot be considered to be homogeneously distributed (and thus interchangeable).

3. SIMULATING CHEMICAL KINETICS WITH DYNAMIC COMPARTMENTS

In this chapter, we present the algorithm developed to simulate the time-evolution of a dynamically compartmentalized chemical reaction system. We implemented these algorithms on top of the SGNS simulator[32]. The resulting program is what was used to simulate the models in section 4.

When considering asymptotic runtimes, R represents the number of different reaction channels in the system, S represents the number of sub-simulations in the NRM (see section 3.2), and M represents the greatest outbound degree in the reaction dependency graph (the graph depicting which propensities must be updated when a given reaction fires). The number of sub-simulations is expected to grow linearly with the number of compartments (section 3.2.2), so S can be considered to be the current number of compartments in the simulation.

Here, we assume that M vanishes with the number of reactions since no reactant is expected to be involved in a non-vanishing number of reactions. It should be noted that there are systems in which this assumption does not hold, such as a gene network where RNA polymerases are involved in all transcription reactions. These cases constitute a weakness in the current implementation, but we consider them to be rare enough to disregard for now. For simplicity, propensity functions $a_\mu(\mathbf{x})$ are henceforth written as a_μ.

3.1 Compartment Definition

The term *compartment* is used here in a loose sense, and does not strictly represent a space which is clearly delineated from its environment, although such compartments can be modeled as well. Compartments group a subset of reactants which interact differently with the rest of the reactants of the simulation. In this sense, compartments act much like different labels used to differentiate subsets of reactants from each other. That is, reactant a in compartment P will react with other molecules in P, but not with other reactants in compartment Q, including other a molecules.

To allow species within compartments to interact with the rest of the species in the system at runtime, we introduce the notion of the compartment *type*. Every compartment is of a single type. Reactions will be specified as occurring within specific compartment types, and when the simulation is actually run, there will be

a separate instance of each of these reactions occurring between reactants in each compartment of the given type. We denote a molecular species a that is 'contained' in a compartment of type P as $a@P$.

Many compartments may exist in the environment in a hierarchy. For example, DNA and RNA exist within the nucleus of a eukaryotic cell, which in turn exists in the environment (which is the cell in this case). We will use this idea of a 'compartment type hierarchy' to construct a well-defined rule-based method to define inter-compartment reactions - reactions will be allowed to span vertically across the hierarchy (between parents and children), but not horizontally (between siblings). In this manner, a subcompartment (a child compartment) can only affect a sibling indirectly by first changing the supercompartment (its parent compartment).

The environment id treated as a special case. When the simulator is initialized, there is a single compartment type, called Env, which represents the environment. There is always a single compartment of type Env, and new Env's cannot be instantiated or destroyed at runtime. Unless otherwise specified, all reactants and reactions occur in Env, and all compartment types are contained in Env, either directly or indirectly.

To illustrate this, consider a detailed model of transcription and translation [23], taking place in a variable size population of prokaryotic cells. The compartment type hierarchy might resemble Figure 3.1.

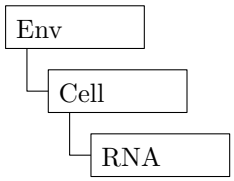

Figure 3.1

Here, the cells are compartments in the normal sense - they have a well-defined border through which molecules must pass before reacting with its contents. Thus, there will be sets of diffusion reactions between the Cell compartment type and Env. The Gene and RNA compartments are not spatially-delineated compartments, and so some reactions will have reactants in two compartments. For example, translation initiation will begin with a ribosome in the Cell compartment binding to the Ribosome Binding Site in the RNA compartment. Since each RNA compartment is distinct from the others, traffic and other effects between elongating ribosomes can be simulated correctly.

3.2 Basic Simulation

We construct the simulation based on a combination of existing algorithms. The overall simulation is controlled by the NRM, since it is an extremely flexible discrete event simulation that can accommodate other simulation algorithms into it. This is accomplished by having the sub-simulations publish a "next firing time" to the NRM, just as the reactions in the NRM have their next reaction time. The use of the indexed priority queue in the NRM allows enough flexibility to add and remove sub-simulations at runtime.

The main simulation loop consists of the following steps:

1. **Selection**: The reaction/event and time of the event to occur next is determined. This step runs in constant time since the NRM stores the next event to occur at the front of the indexed priority queue.

2. **Execution**: The reaction/event is performed, moving time forward and modifying the state of the simulation according to the type of the event that occurred. Reaction propensities that depend on the changed state are flagged as 'dirty'.

3. **Update**: The 'dirty' propensities and timings - those that could affect the timing of the next event - are recalculated and the changes are propagated back up the data structures.

The Execution step takes as much time as the sub-simulation chosen in the Selection step, and the Update step takes $O(\log S \cdot \sum_{S \in dirty} \text{update}(S_i))$, where S is the number of sub-simulations in the system, and update(S) is the runtime of the update step of the dirty sub-simulations. Therefore, for runtime considerations, the execution times of the Execution and Update steps will be considered for each of the sub-simulations.

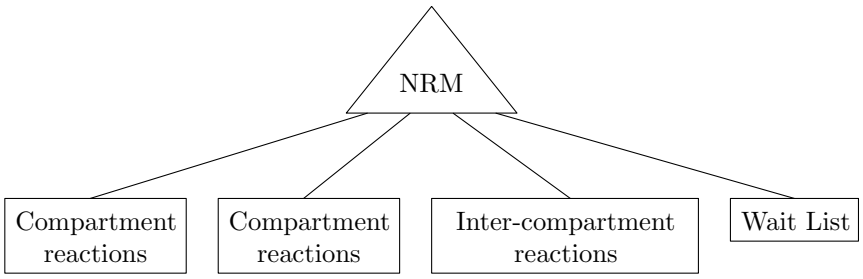

Figure 3.2: A schematic drawing of the structure of the simulator at a point in time where the model has two compartments (Section 3.2.2) at different levels of the hierarchy with at least one vertical reaction between them (Section 3.3.2).

In general, each sub-simulation will be a SSA running in one compartment. In this sense, the NRM integrating the results of all the sub-simulations can be thought

of as a "Next Compartment Method". This setup is reminiscent of the FFM, with the reaction families equal to the sets of reactions that can occur in each compartment. In this case, we are using the NRM to be able to add or remove families of reactions at runtime. Figure 3.2 depicts this construction with some of the sub-simulations that will be introduced in the following sections.

3.2.1 Compartment Representation

Every compartment in the simulation is given its own state vector of species populations. Which species are present in a given compartment's state vector and their layout depend on the compartment's type. This avoids allocating space for species that do not exist in a given compartment. The possible reactions that occur within each compartment are stored with the compartment type, along with the reaction dependency graph which indicates which reactions' propensities must be flagged as dirty when the population of a given molecular species changes.

3.2.2 Simultaneous Simulation of Independent Compartments

Every compartment is given its own instance of a LDM partial sums tree, which will integrate the propensities of all intra-compartment reactions that can occur therein (cross-compartment reactions will be considered later). Each LDM simulation publishes its the time of it next reaction to the overall NRM simulation. The LDM tree is therefore behaving as though it were a pseudo-reaction with propensity a_0.

When a particular compartment has been selected in the Selection step, during the Execution step, the exact reaction to occur is chosen by performing a binary search through the tree of partial sums, as described by [19]. This takes time proportional to the depth of the tree, or $\Theta(\log R)$ time. The reaction is then executed, changing the values in the compartment's state vector. Each changed population flags the dependent reaction propensities as dirty by traversing the reaction dependency graph for the compartment type. Additionally, any delayed products must be inserted into the wait list, an $O(\log R + \log C)$ operation (see section 3.2.3). If N is the number of molecular species whose population changed, and A is the number of delayed products, the total runtime of the Execution step is therefore $O(\log R + N(\log R + \log C) + A \cdot M)$. Since N and A are expected to be constant, and M is expected to vanish with R, the overall runtime of the Execution step is therefore $O(\log R + \log C)$ with wait list additions, and $O(\log R)$ without.

The indices of the dirty propensities are stored in a singly-linked list during the Execution step. To perform the Update step with as little effort as possible, each propensity or partial sum is associated with a counter, C_i indicating the last simulation step on which they were updated. The Update step is then performed as

follows:

repeat
  Let i = the index of the next dirty propensity or partial sum
  Flag propensity i as clean by removing it from the dirty list
  if $C_i <$ current step **then**
    Recalculate the propensity or partial sum i
    Mark the parent partial sum as dirty
    $C_i \leftarrow$ current step
  end if
until no propensities or partial sums are dirty
Randomize or transform (by equation (2.25)) the next reaction time based on the root of the partial sum tree
Publish the new next reaction time to the NRM

Since this algorithm updates M propensities and at most $\log_2 R$ partial sums per updated propensity, its overall runtime is $O(M \log R + \log S)$. Since M is expected to vanish with the system size, the runtime is $O(\log R + \log S)$.

3.2.3 The Wait List

The wait list is a sub-simulation of the NRM, and is implemented as a binary heap. Insertions and removals from the heap take $O(\log W)$, where W is the total number of molecules on the wait list. Since W is expected to grow at most linearly with RC, insertions and removals from the heap can be considered to take $O(\log R + \log C)$ time. The Execution step of the wait list sub-simulation then takes $O(1)$ time since the next wait list event is readily available at the top of the heap, and the Update step takes $O(\log R + \log C)$ time. During another sub-simulation's Execution step, elements may be added to the wait list. This will make the other sub-simulation's Execution runtime to be $O(\log R + \log C)$ as well, provided the number of elements added per step is constant.

To accommodate different compartments, every entry in the wait list includes the compartment in which the molecule is to be released. This has the side effect that, without an additional data structure, the entire wait list must be searched when a compartment is destroyed, an $O(RC)$ operation. Here, we consider compartment construction and destruction as sufficiently rare events to neglect this. One means to solve this problem would be to construct individual wait lists per compartment, segregating the structures that need to be destroyed.

3.3 Inter-Compartment Reactions

The previous section described how intra-compartment reactions are simulated in worst-case logarithmic time per simulation step. This section discusses the implementation of reactions that span between compartments. As mentioned above, horizontal reactions in the compartment hierarchy (those that have reactants or products in two different compartments at the same level in the hierarchy) are presently not allowed.

3.3.1 Simulating Vertical Reactions

To implement vertical reactions, we must first define exactly how such a system should behave. Suppose we have the system of reactions:

$$a@P + b@Q \xrightarrow{c_1} z@Q \tag{3.1}$$

$$b@Q + c@R \xrightarrow{c_2} z@P \tag{3.2}$$

$$a@P + b@Q + c@R \xrightarrow{c_3} z@R \tag{3.3}$$

$$b@Q \xrightarrow{c_4} z@P \tag{3.4}$$

$$a@P \xrightarrow{c_5} z@Q \tag{3.5}$$

where P, Q and R are compartment types and a, b, c and z are molecular species. If Q compartments are contained within a P compartment, and R compartments are contained within Q compartment, these reactions are clearly span vertically across the compartment hierarchy.

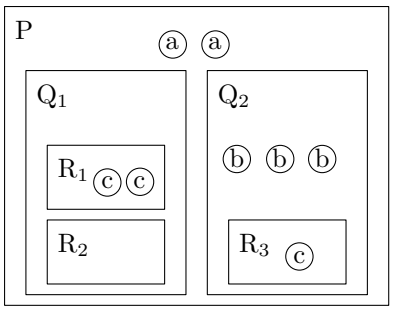

Figure 3.3: An example of the state of the system at some point in time, with compartments shown as rectangles and reactants shown as circles.

Suppose further that the current state of the system is as depicted in Figure 3.3. To simulate this system, we must define a means to calculate the propensity of the above three reactions. In words, reaction (3.1) represents a reaction between an a molecule in a P compartment, and a b molecule in a Q compartment *that is contained*

in the P compartment, which produces a z molecule *in the same Q compartment*. We thus introduce the notion that a particular vertical reaction 'occurs' in the lowest level compartment in which it has a reactant or product. Propensities of vertical reactions are then calculated *only* from the possible combinations of reactants in the compartment the reaction occurs in and its containing compartments. This implies that there is a separate propensity for each lowest-level compartment that a vertical reaction reacts in.

If we extend the homogeneous assumption of the SSA to compartments (that is, Q compartments within P are always homogeneously distributed), we can now write the propensity of the vertical reaction (3.1) in terms of $X_{b@Q}$, the number of b molecules in a given Q compartment, and $X_{a@P}$, the number of a molecules in the P compartment that contains the Q compartment. The propensity a_1 of reaction (3.1) is then:

$$a_1 = c_1 X_{a@P} X_{b@Q}$$

Extending this to reactions (3.2) and (3.3) gives the other propensities as follows:

$$\begin{aligned} a_2 &= c_2 X_{b@Q} X_{c@R} \\ a_3 &= c_3 X_{a@P} X_{b@Q} X_{c@R} \\ a_4 &= c_4 X_{b@Q} \\ a_5 &= c_5 X_{a@P} \end{aligned}$$

Note that reaction (3.2) produces z in a different compartment from any of its reactants. The P compartment is implied to be the compartment containing the Q compartment containing the R compartment in which the reaction occurs.

Reaction (3.1) therefore cannot occur in Q_1, but has propensity $6c_1$ in Q_2. Similarly, reaction (3.2) will not occur in compartments R_1 or R_2, but has propensity $3c_2$ to occur in R_3. Reaction (3.3) can only occur in R_3, with propensity $6c_3$.

A straightforward way to implement vertical reactions would be to include the propensities of each of these reactions with the propensities of the intra-compartment reactions that can occur in each of the compartments. The LDM running in each compartment will then select these reactions to occur with the correct probabilities and in $O(\log R + \log C)$ time. Since multiple reaction propensities in subcompartments depend on a single number (the populations of a molecule in the subcompartments' common supercompartment), a change in the population of the molecule in the supercompartment will cause the LDM update logic to be called once for each subcompartment. This will render the Update step $O(C \log R)$, an unacceptable linear scaling on the number of compartments.

The aforementioned problem only affects vertical reactions with reactants at dif-

ferent levels of the compartment hierarchy. Reaction (3.4) can safely be implemented by adding its propensity to the intra-compartment LDM tree. However, reaction (3.5) cannot since it occurs in the Q compartment, and therefore needs one instance per Q compartment, yet its rate depends on the amount of a molecules in the containing P compartment.

3.3.2 Logarithmic Time Complexity

Some of the reaction systems that this simulator was developed for have large amounts of different compartments in the system at a given time. The linear scaling in section 3.3.1 of the Update step with the number of compartments is therefore expected to have serious performance implications.

Let us consider a compartment P that contains C subcompartments $Q_1 ... Q_C$, between which reaction (3.1) can occur. If X_a is the population of a molecules in compartment P, and X_b^k is the population of b molecules in compartment Q_k, then the propensity a_1^k of the instance of this reaction in the Q_k compartment is equal to $X_a X_b^k c_1$. As in equation (2.13), we can write the total propensity a_1 that *any* instance of this reaction will occur in any Q subcompartment as:

$$a_1 = \sum_{k=1}^{C} a_1^k = \sum_{k=1}^{C} X_a X_b^k c_1 = X_a c_1 \sum_{k=1}^{C} X_b^k \tag{3.6}$$

We observe from (3.6) that a_1 can be factored into two terms: $c_1 X_a$ and $\sum_{k=1}^{C} X_b^k$. If the calculation of a_1 is then broken into two parts, one which calculated $c_1 X_a$ and one which calculates and stores $u_1 = \sum_{k=1} X_b^k$, then a_1 can be readily recalculated with $c_1 X_a u_1$ if X_a changes. We therefore introduce a new data structure: the factored tree of partial sums. This structure resembles the LDM's tree of partial sums, with the propensities of each reaction occurring in the supercompartment as leaves of the tree. We then extend it to include separate trees of partial sums for the calculation of u_i for each reaction.

This data structure is illustrated in Figure 3.4. The illustrated system has four vertical reactions possible between one parent compartment and three subcompartments. In this example, X_a does not specifically represent the number of molecules of a, but the reaction's reactant that exists in the supercompartment. Similarly, X_b^k represents the number of molecules of the reaction's reactant in the subcompartment.

This data structure constitutes a second sub-simulation whose next reaction times are chosen in a similar manner to the LDM sub-simulation (section 3.2.2). The subsimulation is instantiated once for each compartment that can contain subcompartments with which it has spanning multimolecular reactions. Upon the instantiation

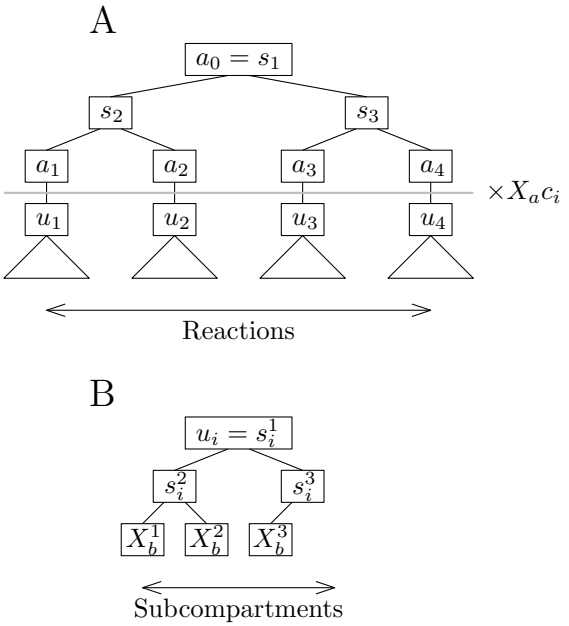

Figure 3.4: Example of the factored partial sums data structure, partway through a simulation with four reactions with reactants in two different compartments. (A) The tree of partial sums calculating the total propensity of all four vertical reactions involving this compartment. The gray line denotes the point where the number of molecules X_a of the reactant in the supercompartment is factored into the final propensity a_i. The partial sums are denoted s_j. (B) Example of the tree of partial sums rooted at each u_i in (A), which calculates part of the total propensity function for reaction $\mathcal{R}_i$.

of a subcompartment which contains reactants that can react with molecules in this compartment, the compartment must be added to all trees of partial sums. The space requirements of this data structure are the same as storing the R propensities in the C intra-compartment partial sum trees: $\Theta(RC)$.

During the Selection step, a binary search is first performed across the tree of partial sums to select which reaction occurs, then the value of Ua_0 is scaled by $X_a^{-1}c_i^{-1}$, and a second binary search is performed to select which compartment the selected reaction occurs in. The amount of work performed is proportional to the sum of the depths of the two trees, $\Theta(\log R + \log C)$.

Since at most $\log_2 R + \log_2 C$ partial sums must be updated for each of M dirty propensities, the runtime of the Update step will be $O(M(\log R + \log C))$. Since the modification of the supercompartment's molecule's population now marks only one position in the tree of partial sums as dirty, M is expected to vanish with R and C, making the final runtime $O(\log R + \log C)$. Trimolecular reactions and above which contain reactants at three or more levels of the compartment hierarchy will require extra trees of partial sums, one for each level of the compartment hierarchy which they have reactants in.

We note that similar runtime bounds can be achieved with a similar factorization scheme constructed with a NRM-like approach. This approach was not pursued here

due to its added complexity. In this scheme, a NRM sub-simulation would contain its own time variable t_P which would advance such that $dt_P = X_a dt$. Care would need to be taken to have the sub-simulation publish its next reaction times in the global simulation time frame which are generated from t_P.

3.4 Dynamic Compartments

Thus far, we have a simulator that is capable of simultaneously simulating the chemical kinetics within and between a fixed set of compartments. This section discusses the implementation of dynamic creation and destruction of compartments.

3.4.1 Compartment Creation

Compartment creation is represented by the following reaction type:

$$\text{reactants} \longrightarrow @Q \tag{3.7}$$

When this reaction occurs, a new compartment of type Q is initialized with all species populations set to 0, and added to the system. Reactants are removed from the respective compartments according to the reaction stoichiometry. A new LDM tree of partial sums is constructed to handle the new compartment's intra-compartment reactions, any factored trees of partial sums from section 3.3.2 involving this compartment are created or expanded to handle vertical reactions. All created sub-simulations then generate their next occurrence time and publish this to the NRM.

The compartment production reaction occurs in the compartment above it. That is, the level of the compartment hierarchy that the $@Q$ product entails is one level above the Q compartment type. If Q compartments are contained in P compartments, then reaction (3.7) would then be considered to be an intra-compartment reaction in P compartments.

The state of the new compartment can be initialized by adding products after the compartment construction product as in:

$$\text{reactants} \longrightarrow @Q + 10a@Q + 2b@Q \tag{3.8}$$

When this reaction occurs, it places 10 a molecules and 2 b molecules in the new Q compartment.

Since the creation of new compartments must initialize every reaction in the new compartment, it takes at least $\Omega(R)$ time. The new sub-simulations must each add themselves to the NRM, each an $O(\log C)$ operation. If V sub-simulations must be added for a given compartment, the compartment creation operation is then

$O(R + V \log C)$. V is constant so long as the depth of the compartment hierarchy is roughly constant with system size, a rather safe assumption, in which case this operation takes $O(R + \log C)$.

3.4.2 Compartment Destruction

Compartment destruction is represented by the following reaction type:

$$@Q \longrightarrow \text{products} \tag{3.9}$$

When this reaction occurs, the Q compartment it occurs in is removed from the system. All sub-simulations associated to the compartment are dismantled and removed from the NRM. Any product on the wait list that targets this compartment must also be removed, implying a linear search through the $O(RC)$ products on the wait list. If V is the number of sub-simulations removed, the runtime is therefore $O(V \log C + RC)$. If a separate wait list is used per compartment, this bound becomes $O(V \log C)$. As with compartment creation, V is expected to be constant with system size, in which case these bounds become $O(RC)$ and $O(\log C)$, respectively.

4. MODELS AND RESULTS

4.1 The λ bacteriophage

The λ bacteriophage behavior upon invading an *E. coli* cell is the most well studied genetic decision circuits[1]. As mentioned in the background, after infecting a cell, a decision takes place between cell death (lysis) and viral dormancy (lysogeny) [29]. During the decision, the genetic circuit of the viral genes processes a number of signals from the cell, the environment, and other viruses in the cell. Arkin and coworkers [1] showed that these factors alone do not explain the outcome of the decision - stochasticity in gene expression also plays a role.

In [45], the authors examined the infection process at the individual phage level with sufficient spatiotemporal resolution to quantify the relevant kinetic parameters and to evaluate the contribution of each parameter in cell-fate heterogeneity. It was discovered that the cell fate decisions were consistent with each phage making an individual decision of which pathway to take, and then having the decisions of all viruses in a single cell integrated in a noise-free manner, such that only a unanimous 'vote' by all viruses leads to lysogeny.

The latter observation implies that, if one is to build an accurate model of this process, one needs to simulate the dynamics of each virus in the cell independently, and then be able to combine their dynamics in the decision making process between lysis and lysogeny. The simulator proposed in this thesis is able to perform this, as it allows the separate simulation of the dynamics of the gene network of each virus independently, and then transportation of the products of gene expression to the cell compartment. The joint decision is then reinterpreted by the viruses. Here, we implement this model to demonstrate that the simulator can model this.

4.1.1 Model

In this model of the λ bacteriophage's decision circuit, we opted to use the modeling approach of [34]. In this strategy, gene expression is modeled as a two-step process, with multiple different delays used on the reaction products to mimic the intermediate steps. The decision itself is performed by the interactions of three genes, Cro, CI and CII[1]. We denote the promoters of these genes as Pro_{xx}, where xx is Cro, CI or CII, and the ribosome binding sites of their mRNA products as RBS_{xx}. We

have three compartment types in this model: the environment, the cell, and the phage. For brevity, here we denote a molecule that is in a cell compartment as $\cdot^C$ and a molecule in a phage compartment as $\cdot^P$. The reactions modeling the gene expression of the three viral genes involved in the decision are:

$$\mathrm{Pro}^P_{RM} + \mathrm{RNAp}^C \xrightarrow{k_t} \mathrm{Pro}^P_{RM}(\tau_p) + \mathrm{RNAp}^C(\tau_{pol}) + \mathrm{RBS}^C_{CI}(\tau_p) \tag{4.1}$$

$$\mathrm{Pro}_{RM} \cdot 2\mathrm{CII}^P + \mathrm{RNAp}^C \xrightarrow{k_t} \mathrm{Pro}_{RM} \cdot 2\mathrm{CII}^P(\tau_p) + \mathrm{RNAp}^C(\tau_{pol}) + \mathrm{RBS}^C_{CI}(\tau_p) \tag{4.2}$$

$$\mathrm{RBS}^C_{\mathrm{CI}} + \mathrm{Rib}^C \xrightarrow{k_{trans}} \mathrm{RBS}^C_{\mathrm{CI}}(\tau_{rbs}) + \mathrm{Rib}^C(\tau_{\mathrm{Rib}}) + \mathrm{CI}^C(\tau_{\mathrm{Rib}}) \tag{4.3}$$

$$\mathrm{Pro}^P_L + \mathrm{RNAp}^C \xrightarrow{k_t} \mathrm{Pro}^P_L(\tau_p) + \mathrm{RNAp}^C(\tau_{pol}) + \mathrm{RBS}^C_{\mathrm{Cro}}(\tau_p) + \mathrm{RBS}^C_{\mathrm{CII}}(\tau_p) \tag{4.4}$$

$$\mathrm{RBS}^C_{\mathrm{Cro}} + \mathrm{Rib}^C \xrightarrow{k_{trans}} \mathrm{RBS}^C_{\mathrm{Cro}}(\tau_{rbs}) + \mathrm{Rib}^C(\tau_{\mathrm{Rib}}) + \mathrm{Cro}^C(\tau_{\mathrm{Rib}}) \tag{4.5}$$

$$\mathrm{RBS}^C_{\mathrm{Cro}} + \mathrm{Rib}^C \xrightarrow{k_{trans}} \mathrm{RBS}^C_{\mathrm{Cro}}(\tau_{rbs}) + \mathrm{Rib}^C(\tau_{\mathrm{Rib}}) + \mathrm{Cro}^C(\tau_{\mathrm{Rib}}) \tag{4.6}$$

The expression products all degrade with constant probability per unit time:

$$\mathrm{RBS}^C_{\mathrm{Cro}} \xrightarrow{k_{rbsd}} \tag{4.7}$$

$$\mathrm{RBS}^C_{\mathrm{CI}} \xrightarrow{k_{rbsd}} \tag{4.8}$$

$$\mathrm{RBS}^C_{\mathrm{CII}} \xrightarrow{k_{rbsd}} \tag{4.9}$$

$$\mathrm{Cro}^C \xrightarrow{k_d^{\mathrm{Cro}}} \tag{4.10}$$

$$\mathrm{CI}^C \xrightarrow{k_d^{\mathrm{CI}}} \tag{4.11}$$

$$\mathrm{CII}^C \xrightarrow{k_d^{\mathrm{CII}}} \tag{4.12}$$

All three protein products dimerize before binding to the viral DNA to regulate downstream genes. To account for this, we modified the probability of these binding reactions to simulate the dimerization. This was accomplished by multiplying their propensity with a hill function on the molecule's population with coefficient 2. These bimolecular reactions also have their propensity scaled by the inverse of the cell length l to account for the cell size effects noted in [39]. The length was modeled as a separate species which starts with a population of 100 and causes the cell to

divide when it reaches 200. The interactions between the genes are as follows:

$$\mathrm{Pro}_{RM}^{P} + \mathrm{Cro}^{C} \underset{k_{unbind}^{\mathrm{Cro}}}{\overset{k_{bind}^{\mathrm{Cro}}/l}{\rightleftarrows}} \mathrm{Pro}_{RM} \cdot 2\mathrm{Cro}^{P} \tag{4.13}$$

$$\mathrm{Pro}_{RM} \cdot 2\mathrm{Cro}^{P} \xrightarrow{k_{d}^{\mathrm{Cro}}} \mathrm{Pro}_{RM}^{P} \tag{4.14}$$

$$\mathrm{Pro}_{L}^{P} + \mathrm{CI}^{C} \underset{k_{unbind}^{\mathrm{CI}}}{\overset{k_{bind}^{\mathrm{CI}}/l}{\rightleftarrows}} \mathrm{Pro}_{L} \cdot 2\mathrm{CI}^{P} \tag{4.15}$$

$$\mathrm{Pro}_{L} \cdot 2\mathrm{CI}^{P} \xrightarrow{k_{d}^{\mathrm{CI}}} \mathrm{Pro}_{L}^{P} \tag{4.16}$$

$$\mathrm{Pro}_{RM}^{P} + \mathrm{CII}^{C} \underset{k_{unbind}^{\mathrm{CII}}}{\overset{k_{bind}^{\mathrm{CII}}/l}{\rightleftarrows}} \mathrm{Pro}_{RM} \cdot 2\mathrm{CII}^{P} \tag{4.17}$$

$$\mathrm{Pro}_{RM} \cdot 2\mathrm{CII}^{P} \xrightarrow{k_{d}^{\mathrm{CII}}} \mathrm{Pro}_{RM}^{P} \tag{4.18}$$

$$\tag{4.19}$$

We model the population of cells with the following reactions:

$$\text{split} : @Cell + l^{*} \longrightarrow @Cell \tag{4.20}$$

$$\mathrm{Cro}^{C^{*}} + @Cell \xrightarrow{k_{\mathrm{lysis}}} 150\mathrm{Phage} \tag{4.21}$$

$$\mathrm{Phage} \xrightarrow{k_{\mathrm{infect}}} @Phage + \mathrm{Pro}_{RM}^{P} + \mathrm{Pro}_{L}^{P} \tag{4.22}$$

$$\xrightarrow{k_{\mathrm{growth}}} l \tag{4.23}$$

These reactions model cell division (4.20) which creates new Cell compartments, lysis (4.21) which destroys Cell compartments, infection (4.22) which creates a new Phage compartment, and cell growth (4.23). Here, reaction (4.20) divides all populations of proteins, phages, and the length binomially between the old and new Cell compartments (denoted as split : @Cell). Its propensity function is also set to be non-zero only when the population of l exceeds 200. Similarly, the propensity of reaction (4.21) is set to be only non-zero when the Cro protein has reached 45 proteins in the cell. This was set to match the levels in [1]. The constants used in this model are shown in Table 4.1.

4.1.2 Results

We first test whether or not the model is able to reproduce the differing decision behavior in small and large cells observed in [39]. To do this, we simulated static populations of cells with lengths randomly chosen in the ranges 100-150 (short cells)

Constant	Value	Constant	Value
k_t	0.05	k_{bind}^{Cro}	0.00025
k_{rbsd}	1/600	$k_{unbind}^{\mathrm{Cro}}$	0.0005
k_d^{Cro}	0.000245	k_{bind}^{CI}	0.00025
k_d^{CI}	0.0007	k_{unbind}^{CI}	0.00005
k_d^{CII}	0.007	k_{bind}^{CII}	0.0025
τ_p	32	$k_{unbind}^{\mathrm{CII}}$	0.0025
τ_{rbs}	1	k_{lysis}	1/1800
τ_{pol}	56	k_{infect}	1/300
τ_{Rib}	25	k_{growth}	1/18

Table 4.1: Reaction constants and timing parameters for the λ-phage model. Reaction constants (k_{xx}) are given in s^{-1} and delays (τ_{xx}) are given in s. Values are set to match [1] and [46].

and 150-200 (long cells). The results of the simulations and the decision behavior measured in [45] are shown in Figure 4.1A. Good agreement is observed between the model's behavior and the measured phage behavior.

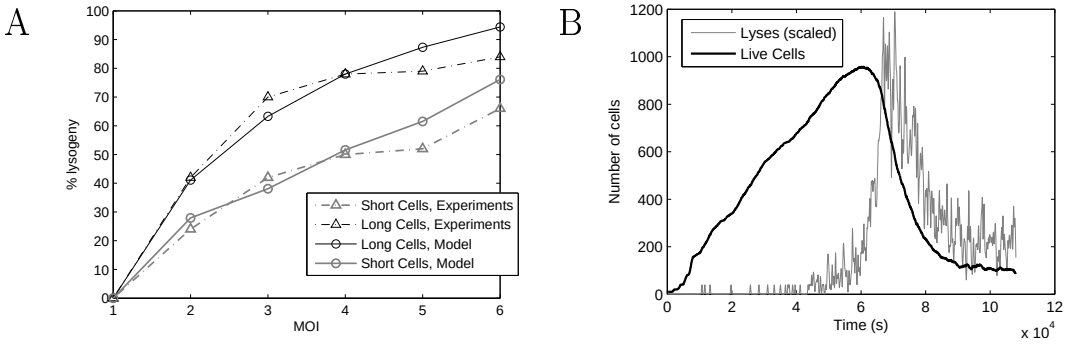

Figure 4.1: (A) The chance of a viral infection taking the lysogenic pathway upon infecting a cell. Measurements from [45] (triangles) are shown with the prediction from the model (circles). Data is shown for short cells ($l < 150$, gray) and long cells ($l \geq 150$, black). (B) Amount of live cells over time in a run of the λ-phage model (black line). The number of lyses occurring over time is also shown (smoothed and scaled for visibility, gray line).

We then simulated the population level model, with an additional restriction: the growth reaction (4.23) was limited by the availability of a molecular species in the environment we called food, which is produced with constant propensity during the simulation. This limiting factor was put in place to avoid an uncontrolled exponential explosion of the number of compartments in the simulation in the case where the phage does not kill all the cells. Figure 4.1B shows a time series of the number of live cells over time. The simulation shows that a stable population of cells can exist even in the presence of the viral infection. We looked for this feature in several different parameter sets and found that the stable population exists for a

large parameter range (data not shown).

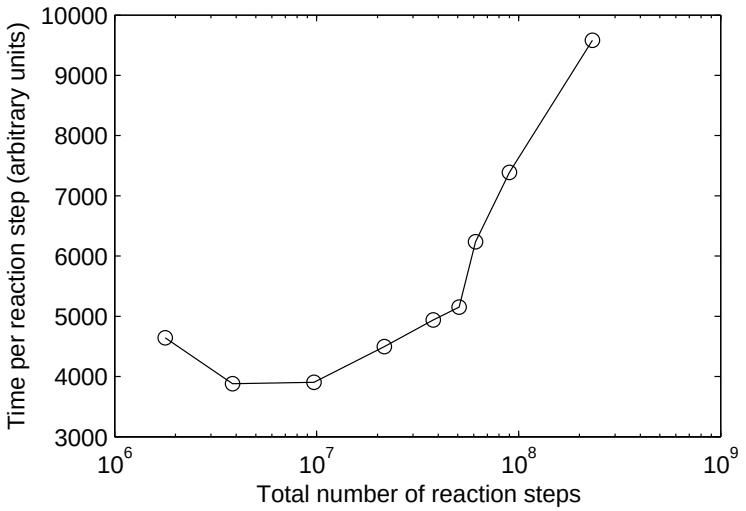

Figure 4.2: Runtime per reaction step against the total number of reaction steps taken by the λ-phage model with varying numbers of cells present in the simulation. Cell counts were varied by changing the mean rate of food production from 0.1 to 50. The $O(\log C)$ scaling of the simulator is visible here, with the simulator taking twice as long per step to simulate models with two orders of magnitude more compartments.

Since this model uses a large amount of dynamic compartments over the course of the simulation, it is useful to test the logarithmic scaling of the simulation with the number of compartments. To vary the number of compartments, we varied the propensity of the food creation reaction, which produced more or less cells (compartments) in the stable population. Since the total number of reaction steps performed should scale roughly as $O(RC)$, when R is held constant, the total number of steps can be used as a proxy for the mean number of compartments in the system at a given time. Figure 4.2 shows the mean time to complete each reaction step against the total number of reaction steps. The trend appears to grow slightly faster than linear on a linear-log plot, consistent with $O(\log C)$ scaling on the number of compartments with a little extra time lost due to $O(R)$ compartment construction and destruction. At the far left of the graph, the simulations complete too quickly, and the initialization time becomes non-negligible, inflating the measured runtime per step.

4.2 Transcription and Translation in Prokaryotes

We simulate a model of transcription first proposed in [33]. This model of transcription is at the nucleotide level and aims to simulate the kinetics of the RNA polymerase as it transcribes a gene. The initial step of transcription initiation is modeled as a delayed event so as to accommodate the kinetics of the closed and open complex

formation. The model additionally includes dynamically relevant events in stepwise elongation such as transcriptional pauses, arrests, error correction, premature termination, pyrophosphorolysis and traffic (that is, collisions between elongating RNA polymerases). Recently, this model was extended [23] to include a single nucleotide model of translation as well. This process is coupled with transcription as it can initiate as soon as the ribosome binding site region of the RNA is produced, and includes events in translation elongation such as ribosomal pauses, back-translocation, rare codons, trans-translation, fall-off and traffic between ribosomes.

4.2.1 Model of Transcription

In this model, each nucleotide (whether DNA or RNA) is represented by several chemical species, each of which always exists in populations of 0 or 1, which describe the various nucleotide's 'state'. The state depends on the status of the activity of an elongating RNA Polymerase (denoted RNAp) or Ribosome (denoted Rib). Unoccupied DNA nucleotides are denoted U_n where n is the nucleotide index in the range $[1, n_{last}]$ and n_{last} is the length of the sequence.

When an RNAp is bound to the DNA and its active site is at the n'th nucleotide, we write the nucleotide's state as O_n. After the RNAp has attached the complementary ribonucleotide to the elongating RNA, it is said to be *activated* and the state is then denoted A_n. Ranges of nucleotides such as $\mathrm{U}_{[start,end]}$ denote all nucleotides in that state with indices from *start* to *end*. Finally, the nucleotide states O_{n_p}, $\mathrm{O}_{n_{ar}}$ and $\mathrm{O}_{n_{\mathrm{correcting}}}$ denote occupied nucleotides when the RNAp is paused, arrested, or error correcting, respectively. The amount of nucleotides from the RNAp's active site that fall under the RNAp as it moves along the DNA (the RNAp 'footprint') is denoted by Δ_{RNAp}.

The reactions modeling transcription are as follows [33]:

$$\mathrm{Pro} + \mathrm{RNAp} \xrightarrow{k_{\mathrm{init}}} \mathrm{RNAp} \cdot \mathrm{Pro}(\tau_{oc}) + @R \tag{4.24}$$

$$\mathrm{RNAp} \cdot \mathrm{Pro} \xrightarrow{k_m} \mathrm{O}_1 + \mathrm{Pro} \tag{4.25}$$

$$\mathrm{A}_n + \mathrm{U}_{n+\Delta_{\mathrm{RNAp}}+1} \xrightarrow{k_m} \mathrm{O}_{n+1} + \mathrm{U}_{n-\Delta_{\mathrm{RNAp}}} + \mathrm{U}^{\mathrm{R}}_{n-\Delta_{\mathrm{RNAp}}} \tag{4.26}$$

$$\mathrm{O}_n \xrightarrow{k_a} \mathrm{A}_n \tag{4.27}$$

$$\mathrm{O}_n \underset{1/\tau_p}{\overset{k_p}{\rightleftharpoons}} \mathrm{O}_{n_p} \tag{4.28}$$

$$\mathrm{O}_{n_p} + \mathrm{A}_{n-2\Delta_{\mathrm{RNAp}}-1} \xrightarrow{0.8k_m} \mathrm{O}_n + \mathrm{A}_{n-2\Delta_-1} \tag{4.29}$$

$$\mathrm{O}_{n_p} + \mathrm{A}_{n-2\Delta_{\mathrm{RNAp}}-1} \xrightarrow{0.2k_m} \mathrm{O}_{n_p} + \mathrm{O}_{n-2\Delta_{\mathrm{RNAp}}-1} \tag{4.30}$$

$$\mathrm{O}_n \underset{1/\tau_{at}}{\overset{k_{ar}}{\rightleftharpoons}} \mathrm{O}_{n_{ar}} \tag{4.31}$$

$$\mathrm{O}_n \underset{1/\tau_c}{\overset{k_{ec}}{\rightleftharpoons}} \mathrm{O}_{n_{\mathrm{correcting}}} \tag{4.32}$$

$$\mathrm{O}_n + @R \xrightarrow{k_{pre}} \mathrm{RNAp} + \mathrm{U}_{\left[n-\Delta_{\mathrm{RNAp}}, n+\Delta_{\mathrm{RNAp}}\right]} \left[\mathrm{Rib}^{\mathrm{R}}\right] \mathrm{Rib} \tag{4.33}$$

$$\mathrm{O}_n + \mathrm{U}_{n-\Delta_{\mathrm{RNAp}}-1} + \mathrm{U}^{\mathrm{R}}_{n-\Delta_{\mathrm{RNAp}}-1} \xrightarrow{k_{pyro}} \mathrm{O}_{n-1} + \mathrm{U}_{n+\Delta_{\mathrm{RNAp}}-1} \tag{4.34}$$

$$\mathrm{A}_{n_{last}} \xrightarrow{k_f} \mathrm{RNAp} + \mathrm{U}_{\left[n_{last}, n_{last}-\Delta_{\mathrm{RNAp}}\right]} \tag{4.35}$$

$$@R \xrightarrow{k_{dr}} \tag{4.36}$$

The reaction constants for these reactions have been chosen to closely match the transcription kinetics of the LacZ gene, and are shown in Table 4.2. These reactions model transcription initiation (4.24), promoter open complex formation (4.25), movement along the template (4.26), elongation (4.27), pausing (4.28), collisions between elongating and paused polymerases (4.29,4.30), arrests (4.31), editing (4.32), premature termination (4.33), pyrophosphorolysis (4.34), completion (4.35) and degradation (4.36). Since there may be elongating ribosomes on the RNA template at the time that reaction (4.33) occurs, and these ribosomes will not be able to complete their proteins since the RNA is incomplete, we 'move' the elongating ribosomes in the RNA compartment ($\mathrm{Rib}^{\mathrm{R}}$) to the environment compartment, denoted as $\left[\mathrm{Rib}^{\mathrm{R}}\right]$ Rib.

The degradation reaction (4.36) actually encompasses several reactions. RNA is degraded when an RNase binds to the RNA and degrades it from the 5' end to the 3' end. This implies that any currently elongating ribosomes on the RNA have a chance to complete their proteins before the RNA is finally degraded. Reaction (4.36) therefore represents several reactions which degrade the nucleotides one at a time and only actually destroy the RNA compartment upon complete degradation.

An example of a simulation of this model is shown in Figure 4.3A. Parameter values were obtained from measurements in *E. coli* for the LacZ gene, since the dynamics of transcription and translation have been extensively studied for it. The LacZ gene has 3072 nucleotides and its transcription is controlled by the lac operon. In this simulation, transcription is not repressed. Thus, provided that the promoter is available for transcription, the expected time for a transcription event to start

Constant	Value	Reference
k_{init}	0.015	[46]
k_m	114	[6]
k_a	114 if $n > 10$	[6]
k_a	30 if $n \leq 10$	[6]
k_p	0.55	[15]
k_{ar}	0.00028	[33]
k_{ec}	0.008	[15]
k_{pre}	0.00019	[18]
k_{pyro}	0.75	[7]
k_f	2	[16]
k_{dr}	0.011	[44]
τ_{oc}	$\sim \mathcal{N}(40, 4)$	[46]
τ_p	3	[15]
τ_{ar}	100	[33]
τ_c	5	[15]

Table 4.2: Reaction constants and timing parameters for the single nucleotide model of transcription of the LacZ gene. Reaction constants (k_{xx}) are given in s^{-1} and delays are given in s. $\mathcal{N}(\mu, \sigma)$ is a normally distributed random number with mean μ and variance σ^2.

is approximately 2.5 s. The promoter open complex formation step, with a mean duration of 40 s [24], is thus the major limiting factor of transcription events in these conditions.

Figure 4.3A shows, for a time window of 400 seconds, the positions (y-axis) over time of several RNAp molecules on the DNA template. Transcription elongation is visibly stochastic, with events such as arrests (for example at $t = \sim 450s$) and ubiquitous pauses. Several collisions between RNAp molecules are also visible, caused in part by these events. Note that one RNAp never overtakes another on the template.

Figure 4.3B shows the distribution of the time intervals between transcription initiation events. This distribution is Gaussian-like due to the open complex formation step. The tail on the right side of the distribution is mainly due to the contribution of the time it takes for the RNAp to bind to the template, since it is a bimolecular reaction with an exponential waiting time with a mean of 2.5 s.

Figure 4.3C shows the distribution of intervals between transcription completion events in the same simulation as Figure 4.3B. This distribution is appreciably different from that of Figure 4.3B due to the stochastic events during transcription elongation. Pauses, arrests and other stochastic events cause the distribution to be bimodal, resulting in bursty dynamics (many short intervals and some long intervals). When these probabilistic events occur to some RNA polymerase molecules, they significantly alter the distances in the strand between consecutive polymerases.

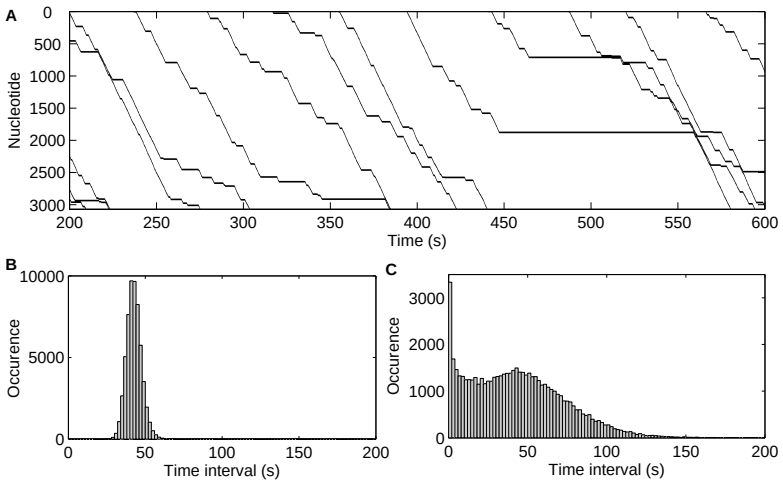

Figure 4.3: Example simulation of the single nucleotide model of the LacZ gene. (A) The progression of multiple RNAps moving along the DNA template. Pauses, arrests and traffic can be seen. (B) Distribution of time between subsequent initiations of transcription. (C) Distribution of time between subsequent completions of transcription. Figure from [23].

For example, when one polymerase pauses, its distance to the preceding one increases, while the distance to subsequent ones shortens, allowing completion events to be separated by intervals shorter than the promoter delay.

To control the level of expression of this model, we introduce the following reaction:

$$\mathrm{Pro} + \mathrm{Rep} \underset{k_{unrep}}{\overset{k_{rep}}{\rightleftharpoons}} \mathrm{Pro} \cdot \mathrm{Rep} \tag{4.37}$$

This reaction models the repression of the promoter by some transcription factor. The level of expression can then be controlled by changing the number of repressors in the simulation.

4.2.2 Model of Translation

In the stepwise model of translation elongation, the transcribed RNA nucleotides are treated in a similar manner to the DNA nucleotides, except that they are placed inside an RNA compartment denoted R. Thus, when transcription is initiated by an RNAp binding to the gene's promoter sequence (Pro) in reaction (4.24), a new RNA compartment is created. $\mathrm{U}_i^{\mathrm{R}}$, $\mathrm{O}_i^{\mathrm{R}}$, and $\mathrm{A}_i^{\mathrm{R}}$ respectively denote unoccupied, occupied and activated ribonucleotides in an RNA compartment. These nucleotides are added to the compartment one at a time by the RNAp in reaction (4.26). When enough ribonucleotides have been transcribed, translation can begin. The single-nucleotide

Reaction Constant	Value	Reference
$k_{\text{trans\_init}}$	0.33	[27]
k_{tm}	1000	[42]
k_{transA}	35	[27]
k_{transB}	8	[27]
k_{transC}	4.5	[27]
k_{bt}	1.5	[36]
k_{drop}	0.000114	[17]
k_{tt}	0.000052	[28]
$k_{\text{trans\_f}}$	2	[27]
k_{fold}	0.0024	[4]
k_{dec}	0.0017	[4]

Table 4.3: Reaction constants (in s) for the single nucleotide model of translation of the LacZ gene and protein.

translation reactions are as follows:

$$\text{Rib} + \text{U}^{\text{R}}_{[1,\Delta_{\text{Rib}}+1]} \xrightarrow{k_{\text{trans\_init}}} \text{O}^{\text{R}}_{1} + \text{Rib}^{R} \tag{4.38}$$

$$\text{A}^{\text{R}}_{n-3} + \text{U}^{\text{R}}_{[n+\Delta_{\text{Rib}}-3,n+\Delta_{\text{Rib}}-1]} \xrightarrow{k_{tm}} \text{O}^{\text{R}}_{n-2} \tag{4.39}$$

$$\text{O}^{\text{R}}_{n-2} \xrightarrow{k_{tm}} \text{O}^{\text{R}}_{n-1} \tag{4.40}$$

$$\text{O}^{\text{R}}_{n-1} \xrightarrow{k_{tm}} \text{O}^{\text{R}}_{n} + \text{U}^{\text{R}}_{[n-\Delta_{\text{Rib}}-2,n-\Delta_{\text{Rib}}]} \tag{4.41}$$

$$\text{O}^{\text{R}}_{n} \xrightarrow{k_{\text{trans\{A,B,C\}}}} \text{A}^{\text{R}}_{n} \tag{4.42}$$

$$\text{O}^{\text{R}}_{n} + \text{U}^{\text{R}}_{[n-\Delta_{\text{Rib}}-2,n-\Delta_{\text{Rib}}]} \xrightarrow{k_{bt}} \text{A}^{\text{R}}_{n-3} + \text{U}^{\text{R}}_{[n+\Delta_{\text{Rib}}-3,n+\Delta_{\text{Rib}}-1]} \tag{4.43}$$

$$\text{O}^{\text{R}}_{n} \xrightarrow{k_{\text{drop}}} \text{Rib} + \text{U}^{\text{R}}_{[n-\Delta_{\text{Rib}},n+\Delta_{\text{Rib}}]} \tag{4.44}$$

$$@R \xrightarrow{k_{tt}} \left[\text{Rib}^{\text{R}}\right] \text{Rib} \tag{4.45}$$

$$\text{A}^{\text{R}}_{n_{last}} \xrightarrow{k_{trans\_f}} \text{Rib} + \text{U}^{\text{R}}_{[n_{last},n_{last}-\Delta_{\text{Rib}}]} + P_{\text{prem}} \tag{4.46}$$

$$P_{\text{prem}} \xrightarrow{k_{fold}} P \tag{4.47}$$

$$P \xrightarrow{k_{dec}} \tag{4.48}$$

The reaction constants for these reactions for the LacZ gene are shown in Table 4.3. The model of translation contains reactions for translation initiation (4.38), 3-step translocation (4.39, 4.40, 4.41), protein elongation (4.42), back-translocation (4.43), premature termination (4.44), trans-translation (4.45), translation completion (4.47), protein folding (4.47) and protein decay or degradation (4.48).

The reaction constants and timings shown in Tables 4.2 and 4.3 are primarily

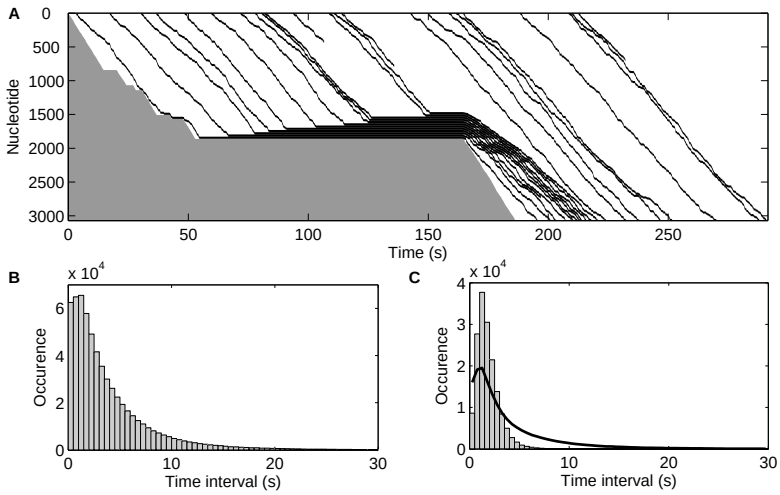

Figure 4.4: Example simulation of the single nucleotide model of the LacZ gene. (A) The progression of multiple ribosomes moving along the RNA template. The gray area denotes the part of the RNA strand that has not yet been transcribed. A long arrest of the RNAp is visible at $t = 50s$, causing ribosomal traffic. (B) Distribution of time intervals between subsequent translation initiations on a single RNA strand. (C) Distribution of time intervals between subsequent translation completions in a model with a long transcriptional arrest at the same time as in (A) (gray bars). The distribution without the transcriptional arrest is shown for comparison (black line). Figure from [23].

for the LacZ gene in *E. coli*. The model of translation, however, contains some parameters which are specific to each codon in the sequence. This allows the model to contain rare codons, which take longer to translate either due to lower amounts of the corresponding transfer RNA (tRNA), or due to the tRNA wobble effect in the third nucleotide[27]. This is encoded by the variable reaction constant of reaction (4.42). Sequence-dependent rarity effects can then be captured by the model.

A time series similar to Figure 4.3A showing ribosomes binding to the RNA as it is being elongated by the RNAp is shown in Figure 4.4A. In this case, a long arrest occurred at $t = 50s$, causing a large amount of ribosomal traffic and illustrating the coupling between transcription and translation in prokaryotes. The distribution of time intervals between subsequent translation initiations is shown in Figure 4.4B, and the corresponding distribution of intervals between subsequent protein completions in the presence of the long transcriptional arrest is shown in Figure 4.4C. When the long transcriptional arrest is present, the distribution of completion intervals differs appreciably.

4.2.3 Results: Dynamics of the model and performance of the simulator

We compared the kinetics of translation of the model to measurements of translation elongation in three engineered *E. coli* strains. These strains were designed to enhance queue formation and traffic in translation [38] by inserting regions of suspected slow-to-translate codons into the gene sequence. The speed of translation elongation was measured by subjecting the cells to a pulse of radioactive methionines, and then measuring the subsequent level of radioactivity incorporated into the produced proteins in the population in the minutes that followed the pulse.

Given the differences between the genes in the three strands, the authors hypothesized that the translation elongation speed of the three strands would differ, since the rate of incorporation of an amino acid depends on which synonymous codon codes for it[38]. The cells where translation is faster will thus be expected to have higher levels of radioactivity in the translated proteins sooner than the cells with the slow-to-translate sequences, as more radiolabeled amino acids have been incorporated in a fixed time interval.

We built the single nucleotide model of each of the three sequences, adding reactions to count the incorporation of radioactive methionines in each protein. We then measured the total incorporation of radioactivity at the same time points after the pulse as in [38]. The results in Figure 4.5A show good agreement between the incorporation predicted by the model with the measurements [23, 38].

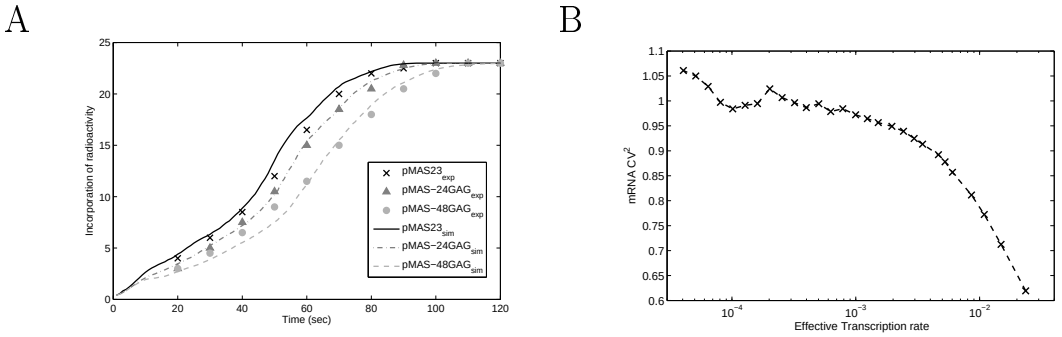

Figure 4.5: (A) Measurements of the incorporation of radioactive methionines into proteins over time (crosses, triangles and circles), and the corresponding predictions of the model (lines). Figure from [23]. (B) Noise (measured by CV^2, the variance over the mean squared) in mRNA levels for varying effective transcription initiation rates. The mRNA degradation rate is set so that the mean mRNA levels at steady state are identical in all cases. Figure from [23].

Next, we investigated the propagation of fluctuations in RNA levels to protein levels. We simulated the model for varying effective rates of transcription initiation. This rate is determined by the basal rate of transcription initiation (k_t), which

determines the binding affinity of the RNAp to the transcription start site, and by the strength of repression of transcription (how often Pro is available for reaction (4.24)). Thus, to vary this rate, we vary the number of repressor molecules present in the system. Three sets of simulations were performed with different rates of translation initiation. In *E. coli* genes, this rate is believed to be determined by the RBS sequence [43]. To study changes in the level of the noise in the absence of changes in mean levels, we fixed the mean RNA and protein numbers at steady state by changing the mRNA and protein degradation rates.

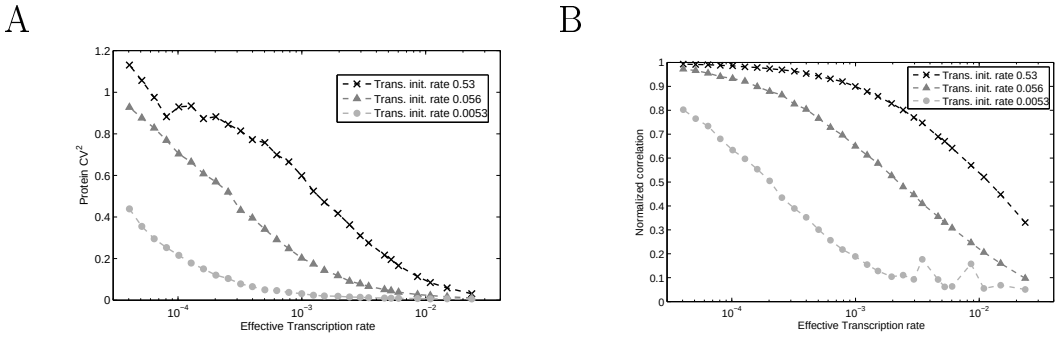

Figure 4.6: (A) Noise (measured by CV^2) in protein levels for varying effective transcription initiation rates and three different rates of translation initiation. mRNA and protein degradation rates are set so that the mean mRNA and mean protein levels at steady state are identical in all cases. Figure from [23]. (B) Normalized maximum correlation between RNA and protein time series. The higher the rate of translation initiation (and thus higher protein degradation to keep the mean the same), the more correlated the fluctuations in protein and RNA levels become, as measured by the normalized maximum correlation. This is because the protein levels follow any fluctuations in the RNA levels faster. Similarly, increasing the rate of transcription initiation, while maintaining the rate of translation initiation constant, decreases the correlation between fluctuations in protein and RNA levels.

For each set of values of the two reaction constants, we performed 100 independent simulations. Depending on these rates, the mean time to reach steady state differs. Each case is simulated for long enough to reach steady state and for an additional 100 000 s after that. The time series of the 100 simulations for each set of parameter values is concatenated into one time series, from which the noise is quantified. This number of long simulations is necessary to properly sample the system due to the stochasticity of the underlying processes.

In Figure 4.5B, the CV^2 of mRNA time series for varying rates of transcription initiation is shown. Noise decreases as this rate increases due to the promoter open complex formation step. Without this event, the distribution of time intervals between transcription initiation events would be exponential, and noise in RNA numbers would not vary significantly. However, with this step, if the expected time for an RNAp to bind to the free promoter is faster than the duration of the

promoter open complex formation, then the distribution of time intervals becomes Gaussian-like.

No measurements have yet been made of the propagation of noise in mRNA levels to protein levels. Nevertheless, it is possible to create a robust estimate, provided reasonable assumptions on the nature of the underlying processes. Our model allows for a direct assessment. Figure 4.6A shows the noise in protein levels, for varying rate of transcription initiation and three rates of translation initiation. The data was obtained from the same simulations used to generate the results in Figure 4.5B.

In general, increasing the rate of transcription initiation decreases the noise in protein levels due to the decrease of noise in mRNA levels. Increasing the rate of translation initiation increases the noise in protein levels. This finding has not yet been experimentally validated by direct means.

From these results, we find that the degree of coupling between transcription and translation is likely to affect the noise in protein levels. This can be verified by computing the normalized maximum correlation between time-series of protein and mRNA levels for each set of parameter values (Figure 4.6B). Comparing this and the previous figure, higher correlation values occur in the regime of higher noise in the protein levels. Therefore, the principal source of this noise is the fluctuations in RNA levels.

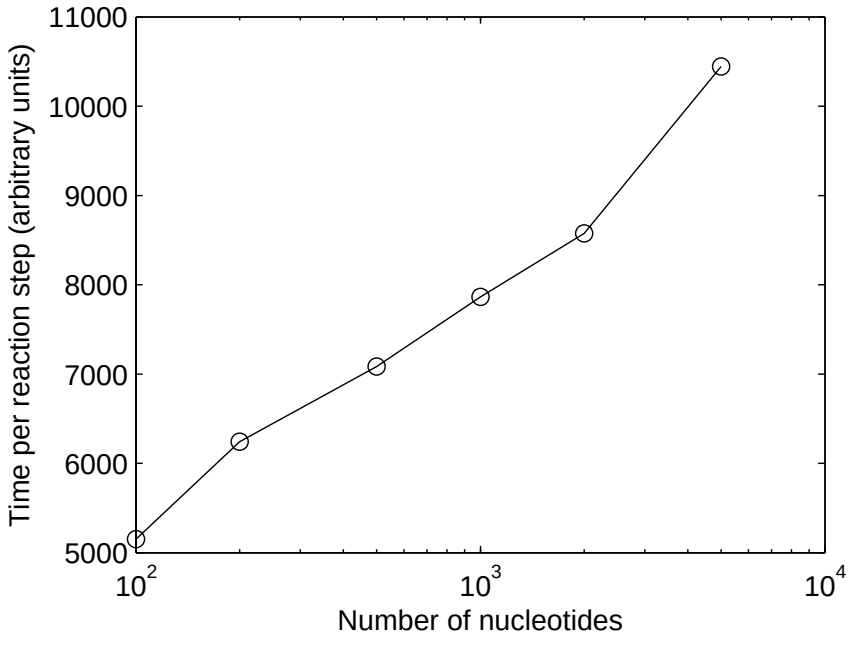

Figure 4.7: Runtime per reaction step against the number of reactions in the single nucleotide model with a varying number of nucleotides. The $O(\log R)$ scaling of the simulator is visible here, as the time per reaction step doubled as the number of reactions was scaled 50 times by changing the number of nucleotides from 100 to 5000 nucleotides.

Given that one of the goals of this thesis is to produce an efficient simulator, we

now turn our attention to testing its efficiency when simulating the single nucleotide model. We did this by simulating the model of random sequences of varying length. Since the model introduces a constant number of reactions per nucleotide, we expect to find a logarithmic dependence between the number of nucleotides simulated and the runtime per reaction step.

From a single run each, we calculated the mean time to complete each reaction step for models with 100 to 5000 nucleotides (Figure 4.7). The result is a straight line on a log-linear plot, indicating that the simulator does scale logarithmically with the number of reactions. We note that strangely, the extra expected runtime per reaction step that is expected to appear due to the compartments remaining in the system longer (they cannot be destroyed until all ribosomes are clear) is not visible in the plot.

The results demonstrate that the simulator is capable of simulating models with a large number of nucleotides in a reasonable timeframe. For example, the simulator running the model with 5000 nucleotides, which contains 83187 reactions, was able to simulate 5.3 million reaction steps in 35 s of real time on a common desktop computer.

5. DISCUSSION

In this thesis we described a new simulator of stochastic chemical kinetics where the spaces in which the chemical reactions occur, and the reactant species therein, can be created and removed during the simulation. This allows the simulation of complex biochemical processes such as the coupled processes of transcription and translation in prokaryotes and the individual decisions by each λ bacteriophage in a bacterium.

The dynamics of the simulated systems are based on stochastic chemical kinetics. Specifically, they are based on the Chemical Master Equation (CME) and a Monte Carlo algorithm to exactly sample the CME, the Stochastic Simulation Algorithm (SSA). The simulator is built upon the most efficient algorithms available for the exact SSA, taking into consideration the nature of the biological systems it is designed to simulate. The overall runtime if the simulation is $O(\log R + \log C)$ per reaction step, where R is the number of reactions in the system and C is the number of compartments. To guarantee this bound in the presence of reactions spanning multiple compartments, a new data structure was introduced, the factored tree of partial sums (Section 3.3.2).

In comparison to previous simulators of stochastic chemical kinetics driven by biological motivations, the new simulator adds in that it allows the creation and destruction of compartments and sub-compartments of reactions at run time. This is critical to, for example, simulate the coupled processes of transcription and translation in bacteria. Without it, the number of reactions and reactants that would need to be explicitly enumerated at the start of the simulations would be prohibitive.

The concept of a dynamically compartmentalized structure for simulating systems of chemical reactions is in itself novel, with many potential applications. It can be used in models ranging from processes within cells, including eukaryotic cells where compartmentalization plays a major role in the cell's function, to dynamic populations of cells with the ability to communicate among them. Some major applications can be easily foreseen. First, from the examples shown, we can see the usefulness of the simulator in modeling gene regulatory networks in prokaryotic cells. Also, one can model populations of cells which can interact with each other as well as the environment (in our example this consisted of a bacterial cell population subject to the λ-phage). These two examples could even be merged to simulate the

dynamics of gene expression of the infecting viruses within each cell, at the single nucleotide and codon levels.

The new features of the simulator would also easily allow the modeling of genetic circuits within cells, each with the ability to perform gene duplication and drift, and subject to environmental selective pressure. As one can model gene expression at the nucleotide and codon level, the present simulator can be used to model evolution at the sequence level. For example, the possible changes in the sequence of the genes (for example, the random deletion of a portion of the sequence, replacement of one promoter by another, addition of a pause-prone sequence, et cetera) could be prestated, and then the cells would be subject to selective pressure by forcing them to compete for limited resources in the environment. Their competitiveness would depend on the gene network's dynamics and would thus be evolvable.

In regard to the two models used as examples, we would like to draw attention to the following specific findings. In the λ-phage model, the most interesting observation is that the virus can reach an equilibrium between lysis and lysogeny, and so a stable cell population in the presence of the virus is possible. Preliminary simulations showed that there is a wide range of parameters in which this equilibrium exists. Future studies will further elucidate the regions of the parameter space in which this equilibrium emerges.

From the simulations of the model of transcription and translation at the nucleotide and codon levels, studied in greater detail in this thesis than the λ-phage model, we were able to characterize and identify the sequence-dependent kinetic features that control the propagation of fluctuations from RNA to protein numbers as well as those that allow the decoupling, from the dynamical point of view, of the time series of numbers of these two molecules which are key in regulating the cellular phenotype. Another important result is that sequence dependent events such as pauses have a non-negligible effect on the dynamics of single gene expression as well as on the dynamics of small genetic circuits. This is of relevance as this study is the first that shows that specific events in transcription elongation can have detectable effects on the fluctuations in protein levels. Since these fluctuations can determine the cell's phenotype, these events may be a source of phenotypic diversity in a monoclonal cell population.

The algorithms presented here provide the basis for a spatial stochastic simulator with a changing number of compartments. Three extensions are readily possible. First, many biological models contain growing or shrinking compartments, such as the λ-phage model presented above. In this case, bimolecular reactions were scaled by the inverse of a pseudo-chemical species l that was created to represent the length of the cell (and thus the cell's volume). The population of the species was then incremented slowly over time. It would be preferable to have the simulator

handle the compartment size without these kinds of hacks. The first challenge is to find a way to efficiently allow the propensities of all bimolecular reactions within a compartment to be scaled by the inverse of the volume V. One readily available method is to re-use the factored tree of partial sums (Section 3.3.2), and factor out the inverse of the volume from all bimolecular reactions. This requires that spontaneous reactions (zero-reactant reactions), unimolecular reactions, bimolecular reactions, and so forth each have their own tree of partial sums, with the appropriate function of the volume factored out at the root. The interactions between this structure and the factoring of the vertical reactions will complicate matters, but should not be an overwhelming obstacle.

Growing/shrinking compartment volumes also pose another difficulty: if they are to be handled exactly, then the firing times of bimolecular reactions in the compartment no longer follow an exponential distribution. To simulate this exactly, the bimolecular reactions within a growing compartment must be placed in a separate tree of partial sums, with the volume factored out at the root as described in the previous section. It should then be possible to derive the distribution of τ, the time until the next bimolecular reaction, given that the propensity function is scaled by a time-dependent variable (V)[8]. Meanwhile, unimolecular reactions have a separate tree of partial sums, which produce the normal exponential waiting times.

Another potential application of the factored tree of partial sums is to efficiently simulate models where there exists a reactant which is involved in a non-vanishing number of reactions. This causes the 'sparse dependency graph' assumption to be violated. For example, in a model of a gene network with N genes, RNAp is involved in $\Theta(N)$ reactions. In this case, the offending reactant could be factored out in the same manner as the supercompartment's reactants. This would allow the simulator to support semi-sparse (where a fixed number of reactants react in a non-vanishing amount of reactions) in worst-case logarithmic time.

A final, and arguably more serious limitation of the current simulation is that it does not support reactions that span horizontally across the compartment hierarchy. The primary reason for this is that we would need a way to specify rules for which compartments the horizontal reactions should occur in (that is, which compartment is each reactant/product in), and the means with which to describe these relations. If this set of rules is given, then the implementation of reactions between sibling compartments follows straightforwardly. That is, a separate LDM tree of partial sums can be created for every set of these horizontal reactions between two compartments. This would allow, for example, a lattice of compartments to be created, with diffusion/interaction between neighboring compartments. Another more complex example might be direct cell-to-cell communication if the cells get close enough in a spatial simulation of the cell's positions.

BIBLIOGRAPHY

[1] A. Arkin and H. H. McAdams J. Ross. Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected *Escherichia coli* cells. *Genetics*, 149:1633–1648, 1998.

[2] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J. Chem. Phys.*, 121:4059–4068, 2004.

[3] H. H. Chang, M. Hemberg, M. Barahona, D. E. Ingber, and S. Huang. Transcriptome-wide noise controls lineage choice in mammalian progenitor cells. *Nature*, 453:544–547, 2008.

[4] B. P. Cormack, R. H. Valdivia, and S. Falkow. Facs-optimized mutants of the green fluorescent protein (gfp). *Gene*, 173(1):33–38, 1996.

[5] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain. Stochastic gene expression in a single cell. *Science*, 297:1183–1186, 2002.

[6] V. Epshtein and E. Nudler. Cooperation between rna polymerase molecules in transcription elongation. *Science*, 300(5620):801–805, 2003.

[7] D. A. Erie, O. Hajiseyedjavadi, M. C. Young, and P. H. von Hippel. Cooperation between rna polymerase molecules in transcription elongation. *Science*, 262:867–873, 1993.

[8] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, 104:1876–1889, 2000.

[9] D. T. Gillespie. A general method for numerically sampling the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.*, 22:403–434, 1976.

[10] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[11] D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188:404–425, 1992.

[12] D. T. Gillespie. The multivariate langevin and fokker-planck equations. *Am. J. Phys.*, 64:1246–1257, 1996.

[13] D. T. Gillespie. The chemical langevin equation. *J. Chem. Phys.*, 113:297–306, 2000.

[14] D. T. Gillespie. Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem.*, 58:35–55, 2007.

[15] S. J. Grieve and P. H. von Hippel. Thinking quantitatively about transcriptional regulation. *Nat. Rev. Mol. Cell. Biol.*, 6:221–232, 2005.

[16] S. J. Grieve, S. E. Weitzel, J. P. Goodarzi, L. J. Main, Z. Pasman, and P. H. von Hippel. Monitoring rna transcription in real time by using surface plasmon resonance. *Proc. Natl. Acad. Sci. USA*, 105:3315–3320, 2008.

[17] F. Jorgensen and C. G. Kurland. Processivity errors of gene expression in *Escherichia coli*. *J. Mol. Biol.*, 215:511–521, 1990.

[18] B. Lewin. *Genes*. Jones and Bartlett Publishers, USA, ix edition, 2008.

[19] H. Li and L. Petzold. Logarithmic direct method for discrete stochastic simulation of chemically reacting systems. *Technical report;* Department of Computer Science, University of California: Santa Barbara, 2006.

[20] L. Lok and R. Brent. Automatic generation of cellular reaction networks with moleculizer 1.0. *Nat. Biotech.*, 23:131–36.

[21] R. Losick and C. Desplan. Stochasticity and cell fate. *Science*, 320:65–68, 2008.

[22] H. Maamar, A. Raj, and D. Dubnau. Noise in gene expression determines cell fate in *Bacillus subtilis*. *Science*, 317:526–529, 2007.

[23] J. Mäkelä, J. Lloyd-Price, O. Yli-Harja, and A. S. Ribeiro. Stochastic sequence-level model of coupled transcription and translation in prokaryotes. *BMC Bioinf.*, 12(1):121, 2011.

[24] W. McClure. Rate-limiting steps in rna chain initiation. *Proc Natl Acad Sci USA*, 77:5634–5638, 1980.

[25] J. M. McCollum, G. D. Peterson, C. D. Cox, M. L. Simpson, and N. F. Samatova. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Comput. Biol. Chem.*, 20(1):39–49, 2006.

[26] D. A. McQuarrie. Stochastic approach to chemical kinetics. *J. Appl. Prob.*, 4:413–478, 1967.

[27] N. Mitarai, K. Sneppen, and S. Pedersen. Ribosome collisions and translation efficiency: optimization by codon usage and mrna destabilization. *J. Mol. Biol.*, 382(1):236–245, 2008.

[28] S. D. Moore and R. T. Sauer. Ribosome rescue: tmrna tagging activity and capacity in *Escherichia coli*. *Mol. Microbiol.*, 58:456–466, 2005.

[29] M. Ptashne. *A Genetic Switch: Phage Lambda Revisited*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, third edition edition, 2004.

[30] T. Rajala, A. Häkkinen, S. Healy, O. Yli-Harja, and A. S. Ribeiro. Effects of transcriptional pausing on gene expression dynamics. *PLoS Comput. Biol.*, 6(3):e1000704, 2010.

[31] A. S. Ribeiro. Stochastic and delayed stochastic models of gene expression and regulation. *Math. Biosci.*, 223:1–11, 2010.

[32] A. S. Ribeiro and J. Lloyd-Price. Sgn sim, a stochastic genetic networks simulator. *Bioinformatics*, (6):777–779, 2007.

[33] A. S. Ribeiro, T. Rajala, O.-P. Smolander, A. Häkkinen, and O. Yli-Harja. Delayed stochastic model of transcription at the single nucleotide level. *J. Comput. Biol.*, 16:539–553, 2009.

[34] A. S. Ribeiro, R. Zhu, and S. A. Kauffman. A general modeling strategy for gene regulatory networks with stochastic dynamics. *J. Comput. Biol.*, 13(9):1630–1639, 2006.

[35] M. R. Roussel and R. Zhu. Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression. *Phys. Biol.*, 3:274–284, 2006.

[36] S. Shoji, S. E. Walker, and K. Fredrick. Ribosomal translocation: One step closer to the molecular mechanism. *ACS Chem. Biol.*, 4:93–107, 2009.

[37] A. Singh and L. S. Weinberger. Stochastic gene expression as a molecular switch for viral latency. *Curr. Opin. Microbiol.*, 12:460–466, 2009.

[38] M. A. Sorensen and S. Pedersen. Absolute in vivo translation rates of individual codons in *Escherichia coli*. *J. Mol. Biol.*, 222:265–280, 1991.

[39] F. St-Pierre and D. Endy. Determination of cell fate selection during phage lambda infection. *Proc. Natl. Acad. Sci. USA*, 105:20705–20710, 2008.

[40] G. M. Suel, R. P. Kulkarni, J. Dworkin, J. Garcia-Ojalvo, and M. B. Elowitz. Tunability and noise dependence in differentiation dynamics. *Science*, 315:1716–1719, 2007.

[41] Y. Taniguchi, P. J. Choi, G.-W. Li, H. Chen, M. Babu, J. Haern, A. Emili, and X. S. Xie. Quantifying *E. coli* proteome and transcriptome with single-molecule sensitivity in single cells. *Science*, 329:533–538, 2010.

[42] J. D. Wen, L. Lancaster, C. Hodges, A. C. Zeri, S. H. Yoshimura, H. F. Noller, C. Bustamante, and I. Tinoco Jr. Following translation by single ribosomes one codon at a time. *Nature*, 452:598–603, 2010.

[43] O. Yarchuk, N. Jacques, J. Guillerez, and M. Dreyfus. Interdependence of translation, transcription and mrna degradation in the lacz gene. *J. Mol. Biol.*, 226:581–596, 1992.

[44] J. Yu, J. Xiao, X. Ren, K. Lao, and X. S. Xie. Probing gene expression in live cells, one protein molecule at a time. *Science*, 311:1600–1603, 2006.

[45] L. Zeng, S. O. Skinner, C. Zong, J. Sippy, M. Feiss, and I. Golding. Decision making at a subcellular level determines the outcome of bacteriophage infection. *Cell*, 141:682–691, 2010.

[46] R. Zhu, A. S. Ribeiro, D. Salahub, and S. A. Kauffman. Studying genetic regulatory networks at the molecular level: delayed reaction stochastic models. *J. Theor. Biol.*, 246:725–745, 2007.